UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

DESENVOLVIMENTO DE UMA REDE SOCIAL DESPORTIVA

Mestrado em Engenharia Informática
Especialização em Engenharia do Software

Rodolfo José Machado Silva

Projeto orientado pelo Prof. Doutor Francisco José Moreira Couto e coorientado por Jorge Filipe Pinheiro Guerra de Ribeiro Teixeira

2016

Agradecimentos

Embora o percurso académico seja individual, nunca poderão ser esquecidas as pessoas que estiveram ao meu lado. Em primeiro lugar quero agradecer aos meus familiares principalmente aos meus pais, tios, avós e ao meu irmão pelo carinho e a apoio que me tem dado ao deste percurso e de toda a minha vida.

Agradeço a todos os meus professores especialmente aos professores Francisco Couto e Jorge Teixeira pela disponibilidade e orientação deste longo projeto de Mestrado.

Agradeço à minha namorada Filipa Ferreira pelo apoio, amor e carinho que no qual tive o privilégio de a ter sempre ao meu lado, durante todo o meu percurso académico.

Agradeço a todos os meus amigos e colegas especialmente à Andreia Teixeira, Ilya Fomichev, Rui Neves, Ana Jardim, André Rosa, Nuno Carreiro, Fábio Guilherme, João Cunha, João Rodrigues e Ruben Pavão pelas gargalhadas, ajuda e amizade durante estes anos da faculdade.

E agradeço a todos, que de uma forma direta ou indireta me ajudaram na realização deste Projeto de Mestrado.

Aos meus pais e avós

Resumo

Hoje em dia segundo o estudo da GSMA Intelligence[1] o número de dispositivos móveis com conexão ativa é superior ao da população mundial, ao ultrapassar os 7,2 mil milhões. Com este estudo podemos efetivamente concluir que os dispositivos móveis vieram ajudar a população mundial no seu dia-a-dia, contando já com milhões de aplicações. O número de aplicações tem vindo a crescer exponencialmente em diversas áreas como o entretenimento, educação, desporto, empresas, utilidades, social etc… As redes sociais também acabam por se tornar responsáveis por este aumento de dispositivos móveis, tendo como enorme vantagem a partilha de informação entre utilizadores.

Esta tese nasce de um problema que reside principalmente em Portugal, relacionado com o decréscimo do número de participantes em diversas modalidades desportivas. Este projeto incide sobre uma aplicação desportiva onde estão registados 804 locais em todo o país podendo ser realizadas 1670 modalidades desportivas.

Devido ao número de dispositivos móveis com diferentes resoluções de ecrãs será necessário que a aplicação tenha uma portabilidade de grande eficácia. Consequentemente a aplicação é responsiva dando a possibilidade de os utilizadores usarem tanto no seu *desktop* como no seu dispositivo móvel. A aplicação é completamente inovadora com características bastante interessantes que se espera que irá convencer a população a aderir ao desporto. Deste modo, estão presentes algumas das funcionalidades como: a fácil e rápida procura de modalidades desportivas dado uma localização, uma lista em conjunto com um mapa de locais onde se possa realizar a modalidade pretendida, possibilidade de criação de equipas para modalidades coletivas e registo de locais por parte de utilizadores, no sentido de aconselhar outros membros a praticar certas modalidades desportivas num determinado local. Estas e muitas outras funcionalidades estão presentes na aplicação, contribuído para uma aplicação inovadora que será capaz de melhorar o dia-a-dia da população, colaborando para uma vida mais saudável e mais ativa. Vários utilizadores foram convidados a interagir com a aplicação e classificaram como "muito fácil de utilizar" e mostraram o seu interesse em usar e partilhar futuramente.

Palavras-chave: Aplicações de desporto, aplicação web de deporto, rede social desportiva, criar equipas, desporto coletivo

[1]GSMA Intelligence "Definite data and analysis for the mobile industry" https://gsmaintelligence.com/ Date: 25/10/2015

Abstract

Nowadays according to the study of GSMA Intelligence[1] the number of mobile devices with active connection is greater than the world's population over 7.2 billion. With this study we can effectively conclude that mobile devices have helped the world's population in their daily lives, having already millions of applications. The number of applications has been growing exponentially in several areas such as entertainment, education, sports, business, utilities, social etc ... Social networking also are responsible for this increase in mobile devices, with the huge benefit sharing information between users.

This project born of a problem that lies mainly in Portugal, related to the decrease in the number of participants in various sports. In this application are recorded 804 locations in Portugal, and can perform 1670 sports. The goal of this project is to build an application that a user can easily combine an event with friends and book a place to practice.

Because of the number of mobile devices with different resolutions screens will require the application have a very effective portability. Consequently the application is responsive giving the possibility of users use both your desktop and on your mobile device. The application is completely innovative with some interesting features that will convince people to join the sport. Therefore we present some of the features as: easy and fast searching of sports given a location, a list and a map of places where you can perform your sport, possibility of creating teams for team sports and recommendation of places by users, to advise other members to practice certain sports. These and many other features are present in the project contributed to an innovative application that will be able to improve the daily life of the population, contributing to a healthier and more active life.

Keywords: *Sports applications, social sport network, create teams, search places to make sport*

[1] GSMA Intelligence "Definite data and analysis for the mobile industry" https://gsmaintelligence.com/ Date: 25/10/2015

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Atualmente em Portugal existe um decréscimo significativo do número de participantes em diversas modalidades desportivas. O problema reside muitas das vezes na dificuldade e no tempo que se gasta em encontrar locais para praticar desporto, ou em caso de modalidades coletivas é por vezes complicado conhecer os interessados em praticar o desporto. Os praticantes também necessitam de contactar diretamente a organização do espaço para saber se está disponível, o que se pode transformar numa barreira à pratica de desporto. Uma notícia[2] no jornal público veio relatar este problema afirmando que Portugal corre o risco de viver apenas de exemplares únicos. As aplicações desportivas já existentes acabam por ajudar uma pequena parte do problema, como é o caso da Nike *Runnning* e *Runkeeper* que vão apresentando o número de km percorridos, a velocidade média e as calorias abatidas, o que acaba por incentivar os praticantes. No entanto, estas aplicações não se adequam a modalidades coletivas e não recomendam locais para praticar desporto. Também já existem aplicações que recomendam locais, mas não permitem que os utilizadores adicionem, comentem, pontuem e formem equipas para combinarem jogos nas diversas modalidades. Para combater esta diminuição de praticantes é necessário estudar estas ideias inovadoras e tentar combinar os pontos fortes das aplicações já existentes e melhorar os pontos fracos para tentar promover a população a aderir ao desporto.

[2] Noticia do Publico - "O desporto português corre o risco de viver apenas de "exemplares únicos" http://www.publico.pt/desporto/noticia/o-desporto-portugues-corre-o-risco-de-viver-apenas-de-exemplares-unicos-169169 Data Consulta: 20/10/2015

1.2 Contextualização do Projeto

Esta tese de Mestrado de Engenharia Informática irá incidir sobre aplicação que será concretizada uma rede social desportiva. Esta aplicação irá complementar e otimizar muitas das componentes existentes em outras aplicações semelhantes e com novas características inovadoras. Estas características irão servir também para melhorar a qualidade de vida da população que gosta de desporto e não o possa realizar por motivos relacionados com a falta de conhecimento sobre determinados locais ou não conheça interessados para realizar uma determinada modalidade desportiva.

A aplicação pretende alcançar o máximo de praticantes e futuros praticantes desportivos que utilizem dipositivos informáticos de forma, aderirem às modalidades desportivas. Desta forma, irá ajudar o crescimento desportivo em termos de número de praticantes e apoiando os futuros praticantes a conhecer as modalidades e os locais que futuramente iram desfrutar, contribuído assim para uma vida mais ativa e mais saudável.

1.3 Objetivos

Com base na problemática relativa ao decréscimo do número de participantes, este projeto tem como objetivo o desenvolvimento de uma rede social desportiva.

Esta aplicação irá ajudar os utilizadores a combinarem um evento desportivo. Desta forma, o utilizador poderá criar uma equipa e receber sugestões de utilizadores com base na distância e nas modalidades desportivas que um determinado utilizador segue. Mais tarde pode criar um evento, convidar os elementos da equipa que pretender e reservar um local para a pratica desse desporto.

Outro objetivo do projeto é a fase de recolha de dados que incide essencialmente na pesquisa de infraestruturas desportivas em Portugal. É essencial procurar dados como contatos, fotografias, localização e modalidades que podem ser praticadas.

Depois da aplicação estar desenvolvida é fundamental haver uma fase de testes com utilizadores para avaliar a usabilidade da aplicação.

1.4 Contribuições

As contribuições para este Projeto de Mestrado são:

1. Implementação da rede social desportiva Sport4Stars. [www.lasige.di.fc.ul.pt/webtools/OlympicSports/]
2. Desenvolvimento da rede social desportiva Sport4Stars para dipositivos moveis.
3. Recolha de dados de 804 infraestruturas desportivas em Portugal.
4. Realização de testes de usabilidade onde os utilizadores foram convidados a interagir com a aplicação.

1.5 Estrutura do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 (Trabalho Relacionado) – Apresenta o estado da arte de ferramentas para o desenvolvimento da aplicação e casos de estudo sobre aplicações semelhantes.
- Capítulo 3 (Análise e Especificação de Requisitos) – Apresenta a análise das funcionalidades que foram desenvolvidas em conjunto com os diagramas de casos de uso e esboços das interfaces da aplicação e a calendarização detalhada do projeto em conjunto com as explicações feitas em relação ao plano inicial do projeto.
- Capítulo 4 (Implementação da Aplicação *Web*) – Apresenta uma descrição da arquitetura utilizada na aplicação e a explicação das funcionalidades mais importantes na aplicação.
- Capítulo 5 (Testes de Usabilidade) – Apresenta os testes de usabilidade realizados para obter resultados relativamente ao uso da aplicação *web.*
- Capítulo 6 (Conclusão) – Apresenta as conclusões finais do projeto.

Capítulo 2

Trabalho Relacionado

2.1 Tecnologias Utilizadas

Neste subcapítulo serão apresentadas as várias tecnologias e bibliotecas utilizadas para a realização da aplicação.

2.1.1 CodeIgniter

O *CodeIgniter*[3] é uma *framework opensource* de PHP que foi desenvolvido sobre o paradigma da programação Orientada a Objetos. A *framework* está organizada num conjunto de *libraries* que estão organizadas numa arquitetura de *design* de forma, a fornecer velocidade, precisão, conveniência e consistência no desenvolvimento de aplicações. As subsecções seguintes foram escritas com base das referências: [1], [2] e [3].

O *CodeIgniter* utiliza o padrão de arquitetura de *software* Model-View-Controller (MVC) contribuindo para uma melhor estruturação de código.

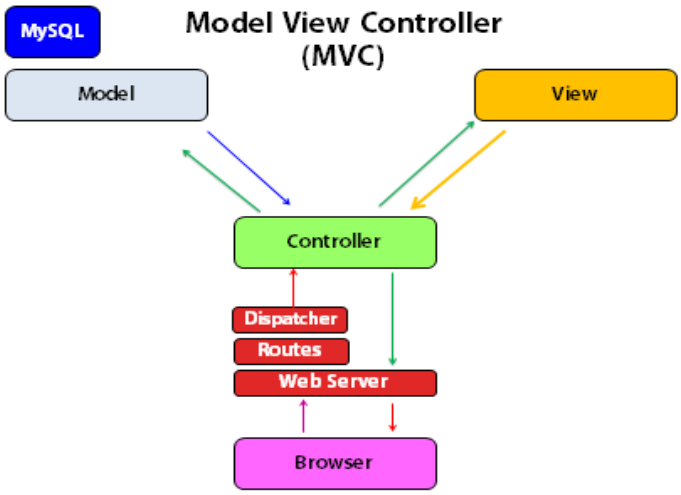

Figura 2.1 Padrão *Model View Controller*

[3]www.codeigniter.com

Analisando a Figura 2.1 a estrutura de trabalho começa a partir do *Browser* que interage como o *Controller* onde irá receber e responder a todas as solicitações do *Browser*. Assim que o *Browser* solicita uma página web o *Router* irá encontrar o *Controller* que será responsável por tratar o pedido, solicitando ao *Model* para aceder os dados e à *View* os exibir.

Model

A principal função do *Model* é a manipulação e recuperação dos dados da base de dados.

Na Figura 2.2 é apresentado uma função *insertUser* que vai inserir um novo utilizador.

```
public function insertUser($id, $name)
{
    $data = array(
        'id' => $id,
        'name' => $name,
    );
    $this->db->insert('user', $data);
}
```

Figura 2.2 - Exemplo do *insert*

Na Figura 2.3 é apresentado uma função que será responsável por consultar a informação de um dado utilizador

```
public function getUser($id)
{
    $this->db->where('id', $id);
    return $this->db->get('user');
}
```

Figura 2.3 - Exemplo do *select*

Na Figura 2.4 é apresentado uma função que será responsável por apagar um dado utilizador

```
public function deleteUser($id)
{
    $this->db->where('id', $id);
    $this->db->delete('user');
}
```

Figura 2.4 - Exemplo de um *delete*

Na Figura 2.5 é apresentado uma função será responsável por atualizar um email de um dado utilizador

```php
public function updateEmailUser($id,$email)
{
    $data = array(
        'email' => $email
    );
    $this->db->where('id', $id);
    $this->db->update('user', $data);
}
```

Figura 2.5 - Exemplo de um *update*

View

Tem como função de transmitir a atividade da consistência de exibição dos dados para as mudanças que ocorrem. Ao agrupar todo o código de apresentação, será mais simples alterar a aparência sem afetar a lógica de negócio e dados. Na Figura 2.6 é apresentado o código de uma *view*.

```html
<div class="list has-head">
    <div class="item item-divider background-separator text-center">
        <span class="text-color-white">Informação do Treino</span>
    </div>
    <div class="list">
        <label class="item item-input">
            <input name="name" id="name" type="text" maxlength="20" placeholder="Nome do Jogo">
        </label>
        <label class="item item-input">
            <input name="date" id="date" class="date_game" id="date_hour_start" placeholder="Data e Hora do Jogo">
        </label>
        <label class="item item-input">
            <input name="duration" id="duration" class="date_game" placeholder="Hora de Terminar" data-view="Dropdown">
        </label>
    </div>
</div>
```

Figura 2.6 - Exemplo de *View* (HTML)

Controller

O controlador define o comportamento que ocorre na aplicação, e em seguida, mapeá-los em função do utilizador para o modelo. No controlador haverá um conjunto de métodos que respondem ao comportamento da aplicação. Na Figura 2.7 é apresentado um exemplo de uma função do *controller*.

```php
public function index()
{
    $id = $this->session->userdata('idUser');
    $data = array(
        'title' => 'Login Page',
        'user' => $this->user_model->getUser($id),
    );
    $this->load->view('index_page', $data);
}
```

Figura 2.7 - Função do *Controller*

2.1.2 Jquery

Jquery é uma biblioteca JavaScript cross-browser *opensource* cujo foi projetado para a criação de scripts no lado do cliente. A sua sintaxe torna a mais simples a seleção de elementos *Document Object Model* (DOM), manipular eventos e desenvolver aplicações AJAX. Assim a sua utilização irá reduzir substancialmente o código em comparação com o JavaScript. Esta secção foi escrita com base na referencia [4].

O DOM é organizado como uma árvore, à semelhança do que acontece no XML. É uma representação hierárquica de uma página HTML, em que cada *tag* do HTML é equivalente a um nó num documento XML. Com o JavaScript e JQuery é possível manipular os elementos HTML que se encontram no DOM. Cada elemento no HTML pode estar identificado por um "id" único.

```javascript
document.getElementById('input').value = 5;
```

Figura 2.8 - JavaScript puro para a atribuição do numero 5 no elemento input

```javascript
$('#input').val( 5 );
```

Figura 2.9 - O mesmo codigo usando Jquery

2.1.3 Cascading Style Sheets (CSS)

O CSS é uma linguagem de folha de estilo usada para definir a apresentação de documentos em uma linguagem de marcação. Junto com o HTML o CSS é uma tecnologia usada em maior parte dos websites para criar paginas web para melhorar o design gráfico. Foi projetado para separar o conteúdo do documento da apresentação do documento, incluindo aspetos de *layout*, cores e fontes. As regras podem ser aplicadas a um elemento individual com um atributo "id" único, a um grupo de elementos que partilham o atributo "class", ou elementos representados pela mesma *tag* HTML.

2.1.4 Asynchronous Javascript and XML (AJAX)

AJAX descreve um modo para realizar uma comunicação de HTTP a partir de um programa JavaScript embutido em uma página web. Esta secção foi baseada na seguinte referencia [5].

O Javascript em execução dentro do *browser* pode atualizar o conteúdo da página web sem a necessidade de descartar a página inteira em cada transmissão de dados. Comunicação acontece de forma assíncrona em segundo plano, a aplicação web permanece totalmente funcional.

A comunicação com um servidor web em AJAX é controlada assim pela chamada XMLHttpRequest, que oferece recuperação de dados assíncrona. No último caso, o pedido é notificado através de chamadas de retorno, quando a recuperação de dados está disponível. O objeto XMLHttpRequest é uma API acessível a partir das principais scripting línguas, como por exemplo JavaScript. Seu ambiente de tempo de execução é o motor Ajax que está sendo incorporado em um padrão web browser.

Algumas aplicações web bastante famosas são utilizadas com esta tecnologia, como por exemplo Google Maps[4] onde os utilizadores podem perfeitamente navegar por mapas sem de que a pagina esteja sempre a fazer *reload*.

[4]https://www.google.pt/maps

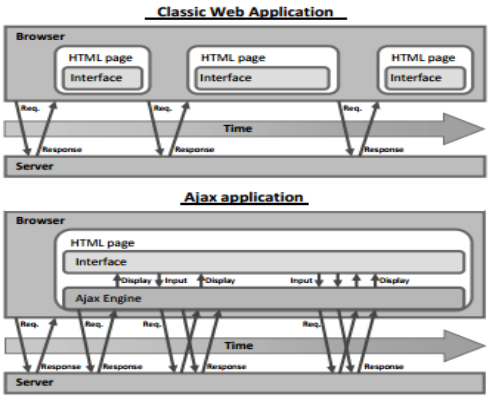

Figura 2.10 - Controlo do fluxo[5] de um site com AJAX em comparação com uma aplicação regular

Na Figura 2.10 na "Classic Web Application" o utilizador faz um pedido HTTP ao servidor web, em que este processa o pedido e retorna uma página HTML ao cliente. Os pedidos adicionais são colocados em espera até que a aplicação atualize a página. Na "Ajax Application" corresponde a uma aplicação web que usa a tecnologia Ajax. Estas aplicações criam um motor baseado em JavaScript que é executado no browser. O motor intercepta inputs do utilizador, exibe o material pedido, e lida com muitas interações no lado do cliente. Se o motor necessitar de mais dados, este solicita material a partir do servidor em background, permitindo ao utilizador continuar a interagir com a aplicação.

2.1.5 Ionic

O *ionic*[5] uma *framework open-source* utilizada no *front-end* para ajudar os programadores a desenvolverem aplicações web responsivas. Utilizando tecnologias como CSS e HTML5, permite um desenvolvimento de bom *design* gráfico em que poderá correr em qualquer dispositivo móvel. Na Figura 2.11 é apresentado o código para formar a *grid* e nas figuras Figura 2.12 e Figura 2.13 é apresentado o resultado em desktop e em mobile.

```
<div class="row responsive-sm">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

Figura 2.11 - Formação da *Grid*

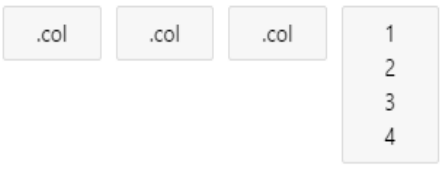

Figura 2.12 - Utilização da *Grid* em desktop

[5]www.ionicframework.com/

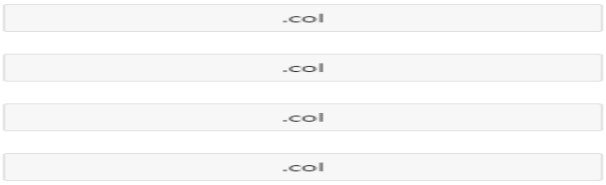

Figura 2.13 - *Grid* em *smartphone*

2.2 Bibliotecas usadas no desenvolvimento do *front-end*

Nesta secção serão apresentadas um conjunto de bibliotecas utilizadas para auxiliar o desenvolvimento do *front-end.*

2.2.1 *Biblioteca DateTimePicker*

Esta biblioteca[6] permite ao utilizador selecionar uma data e uma hora para um determinado evento ou para procurar uma disponibilidade de um local. Para fazer aparecer o *dateTimePicker* basta o utilizador clicar numa caixa de texto ou num *icon.* Na Figura 2.14 apresenta a biblioteca.

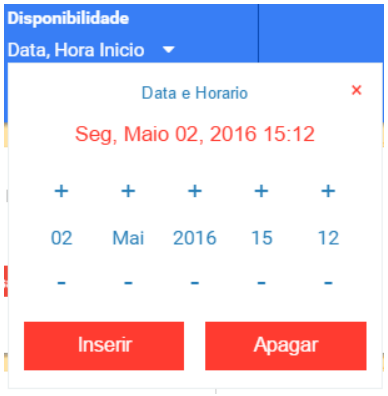

Figura 2.14- Biblioteca *dateTimePicker*

2.2.2 Biblioteca *Ionic checkbox*

Esta biblioteca permite alterar o estilo das *checkbox's* tornando o seu *design* mais interativo e agradável. A sua utilização nesta aplicação foi fundamental principalmente na para a escolha de modalidades. Na Figura 2.15 apresenta a biblioteca utilizada para escolher as modalidades que o utilizador pretende seguir.

[6]https://github.com/CuriousSolutions/DateTimePicker Data Consulta: 10/12/2015

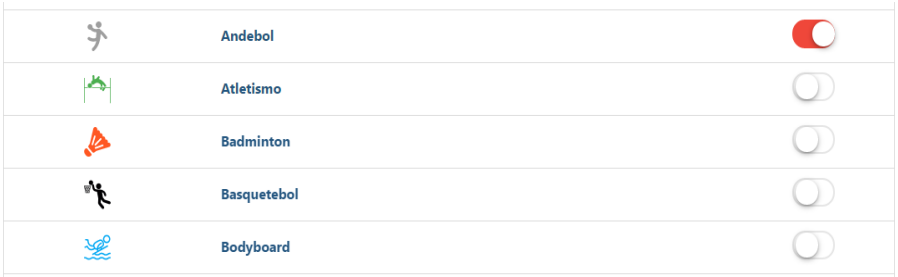

Figura 2.15 - Biblioteca escolha de modalidades

2.2.3 Biblioteca *Ionic Range*

Esta biblioteca é ótima para criar formas interativas de fazer um campo de um formulário que contenha um intervalo de valores. Nesta aplicação foi feito para restringir a distância na procura de locais. Na Figura 2.16 é apresentado a biblioteca range.

Figura 2.16 - Biblioteca ionic range

2.2.4 Biblioteca *rating selector*

Esta biblioteca[7] é um tipo de *checkbox* com a possibilidade de serem inseridas imagens a substituir uma *label* permitindo tornar a interatividade na ação de avaliar um local, mais agradável e acessível. Na Figura 2.17 apresenta a biblioteca em funcionamento.

[7]http://www.jqueryrain.com/demo/jquery-rating-plugin/ Data Consulta: 20/01/2016

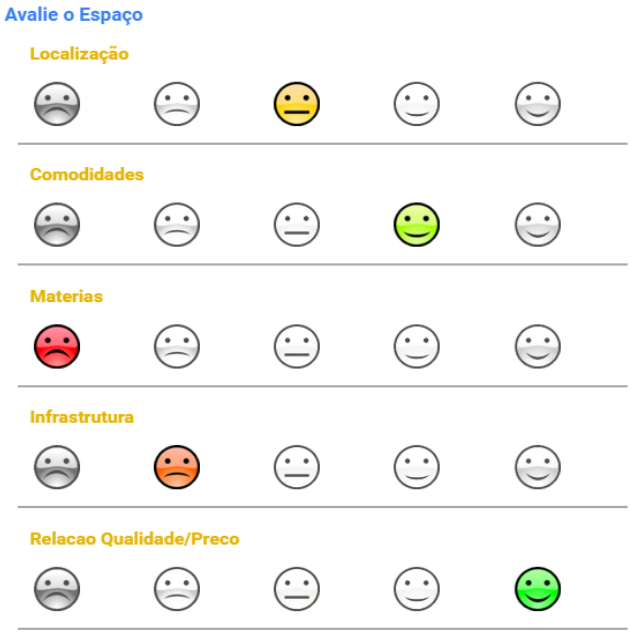

Figura 2.17 - Biblioteca *rating selector*

2.3 Casos de estudo

Nesta secção serão apresentados os vários estudos de aplicações semelhantes.

2.3.1 Onde e quem vai ver

Esta aplicação web foi desenvolvida com o objetivo de facilitar a pesquisa e a divulgação de eventos (por exemplo, eventos culturais, desportivos, musicais, etc…) a realizar em Portugal. Foram desenvolvidas várias funcionalidades como a visualização dos eventos no mapa associado com os filtros de pesquisa, sugestões de eventos, saber que pessoas vão aos eventos (“quem vai?"), convidar amigos para eventos, a criação de eventos, a divulgação de eventos no Facebook, seguir organizadores, a criação do novo conceito “Eu vou Condicional", entre outras. As funcionalidades de saber que amigos vão a um evento (“quem vai?") e partilha de eventos, foram implementadas com base no perfil/lista de amigos do utilizador do Facebook. Na Figura 2.18 apresenta uma página da aplicação para procurar eventos. Esta secção foi escrita com base na referencia [4]

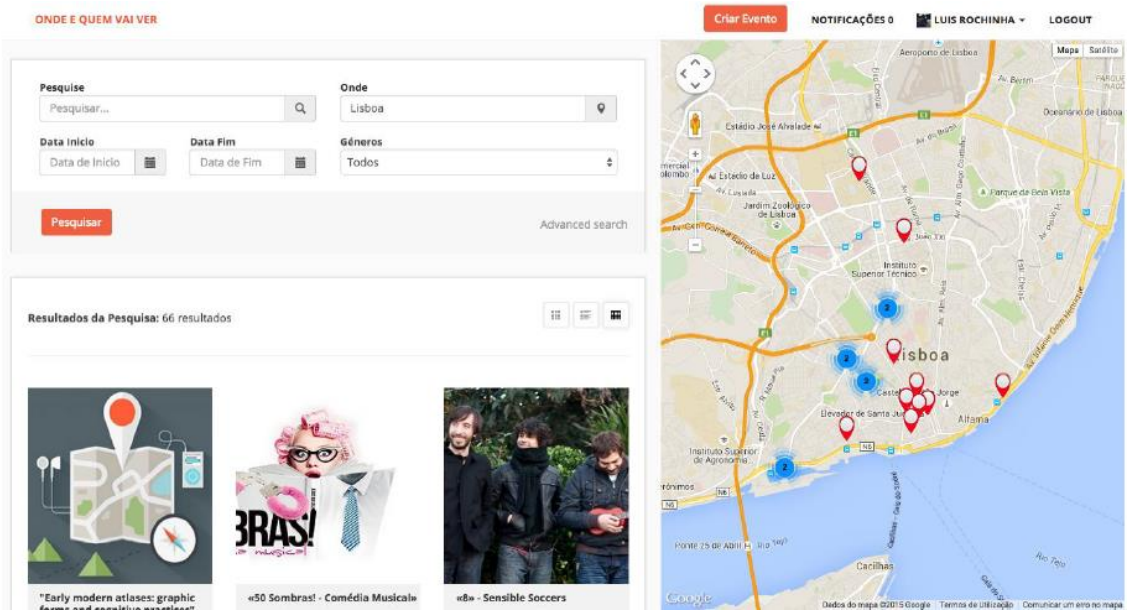

Figura 2.18 - Screenshot Onde e Quem Vai Ver

2.3.2 SportsFinder

A Sports Finder[8] é uma aplicação para dispositivos *android* em que o seu objetivo é encontrar modalidades desportivas com base no código postal, categoria desejada e Desportos. De seguida, devolve uma lista de todos os eventos desportivos nas proximidades, dando a possibilidade de adicionar ao calendário do telemóvel pessoal.

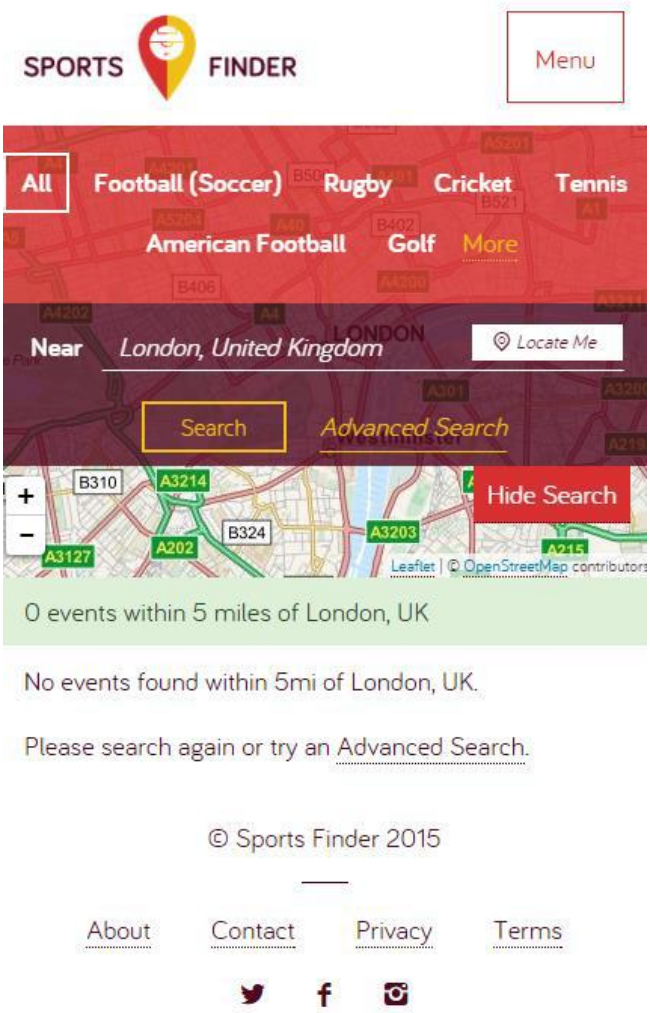

Figura 2.19 - Screenshot Sports Finder

[8]GooglePlay – Sports Finder App
https://play.google.com/store/apps/details?id=com.sim.Hiral.SportEventSearch Data Consulta: 19/10/2015

2.3.3 Nike Futebol

A Nike Futebol[9] é uma aplicação móvel disponível para *smartphones* e *tablets*, cujo objetivo construir equipas de futebol para disputar um jogo num certo local [5]. Esta aplicação tem diversas funcionalidades como:

- Listar eventos desportivos nas proximidades por categoria, fornecidos pelo active.com
- Instruções detalhadas sobre o evento e onde está acontecer.
- Adicionar qualquer número de eventos ao calendário do telemóvel.
- Partilhar eventos entre amigos.
- Participar em jogos perto do utilizador.
- Marcar jogos escolhendo horário e o local.
- Visualizar no mapa os jogos que estão acontecer.

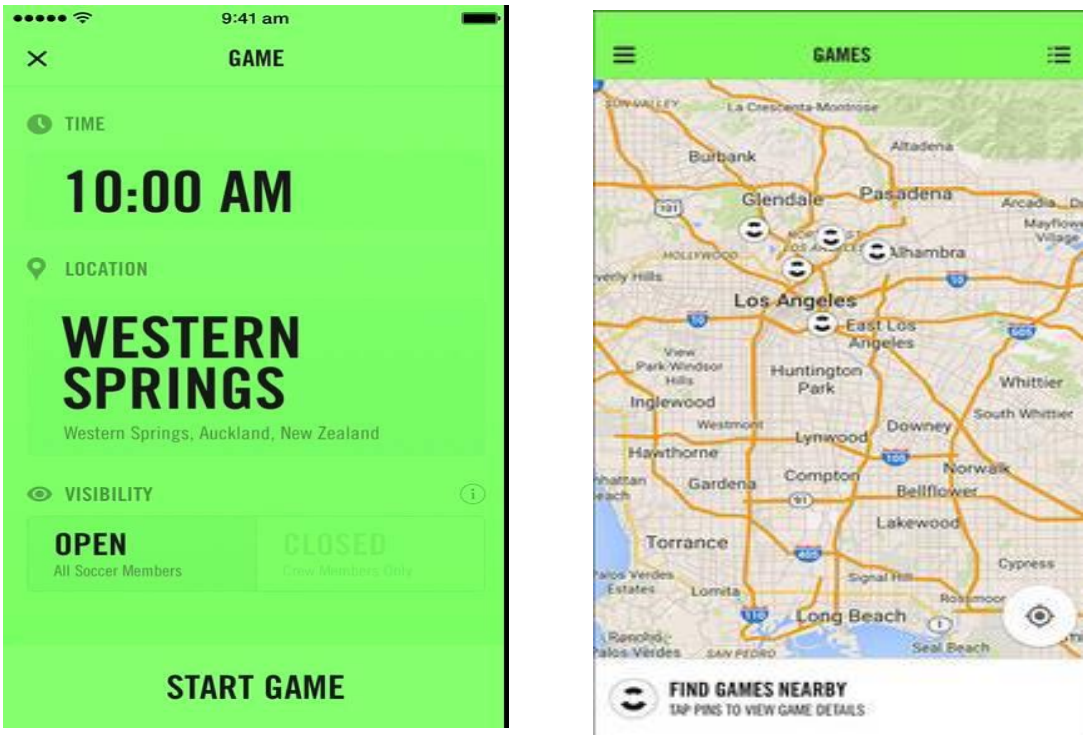

Figura 2.20 - Screenshots Nike Futebol

[9]Google Play – Nike Futebol
https://play.google.com/store/apps/details?id=com.nike.pass.root&hl=pt- PT Data Consulta: 19/10/2015

2.3.4 Playnify

A Playnify[10] é uma rede social para desportistas disponível para smartphones, tablets e desktops. Esta aplicação tem diversas funcionalidades como:

- Criar equipas e combinar eventos num certo local,
- Encontrar instalações desportivas,
- Registar locais,
- Visualizar jogos perto do local.

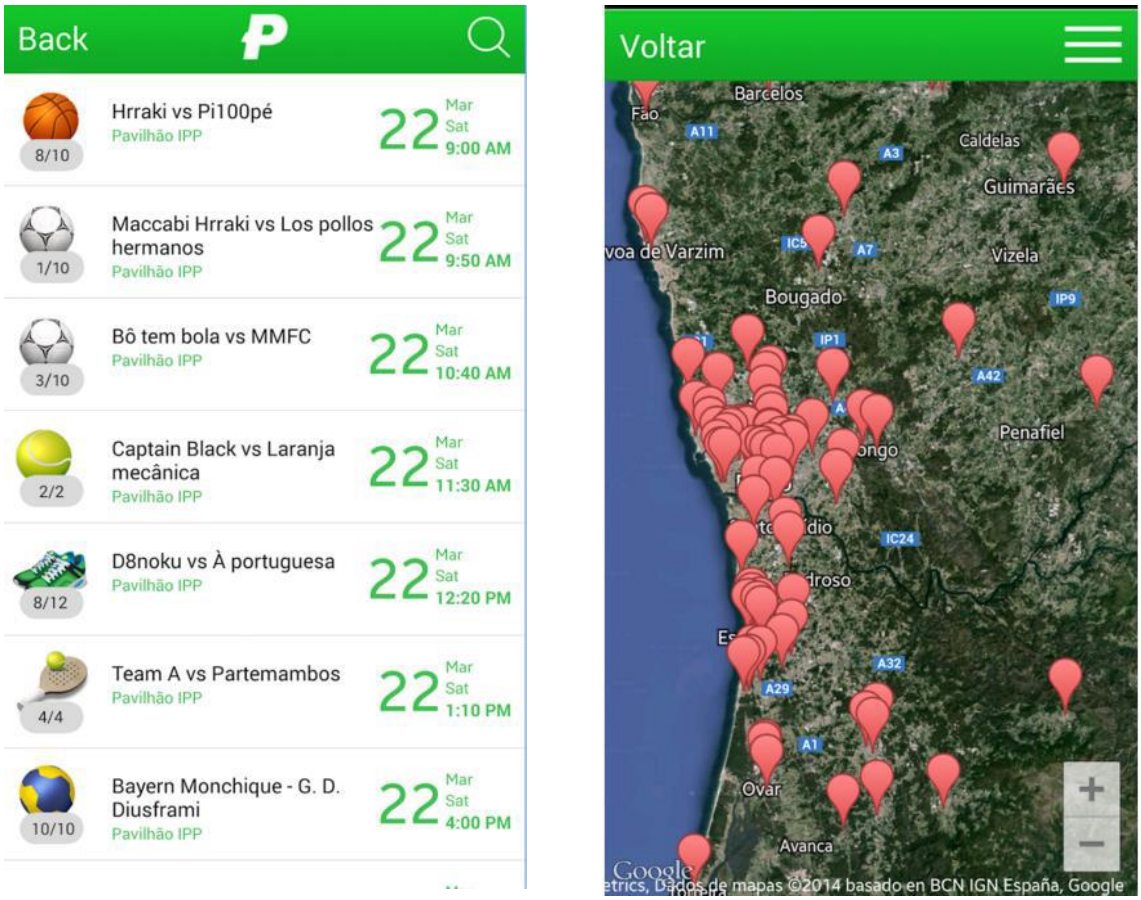

Figura 2.21 - Screenshots Playnify

[10]Playnify– https://play.google.com/store/apps/details?id=com.playnify Data Consulta: 25/10/2015

Capítulo 3

Análise e especificação de requisitos

3.1 Requisitos funcionais e não funcionais

Nesta secção será realizada uma apresentação dos requisitos funcionais e não funcionais.

3.1.1 Requisitos funcionais

Os requisitos funcionais especificam as funções da aplicação. O requisito é o conjunto de *inputs*, de comportamentos e a *outputs*. Na Tabela 3.1 serão apresentados os requisitos funcionais da aplicação web.

Requisito	Descrição
Autenticação e registo através aplicação	Permite a realização de um registo com *username,* email e *password* e login na aplicação com *username* ou email e a respetiva password.
Autenticação e registo através do Facebook	Permite a realização do registo e login na aplicação através do Facebook.
Seguir modalidade	Permite ao utilizador seguir diversas modalidades para receber sugestões de equipas e amigos que sigam essa modalidade.
Visualização de sugestões do utilizador	Permite ao utilizador receber sugestões de equipas e amigos para adicionar, com base nas modalidades seguidas e com base na localização que o utilizador se encontra em comparação à da equipa ou amigo.

Visualizar equipas	Permite ao utilizador visualizar as equipas que o utilizador é membro.
Visualizar jogos	Permite ao utilizador visualizar os jogos que o utilizador irá disputar nas diversas equipas.
Visualizar resultados	Permite ao utilizador visualizar os resultados dos duelos realizados.
Visualizar amigos	Permite ao utilizador visualizar os amigos adicionados.
Visualizar os cometários dos locais	Permite ao utilizador visualizar os comentários que escreveu sobre os determinados locais
Visualizar perfil de utilizadores	Permite visualizar o perfil de um determinado utilizador e enviar um pedido de amizade.
Alteração de dados do perfil	Permite realizar a alteração dos dados do perfil como primeiro e ultimo nome, email, *username*, localização, modalidades seguidas e utilizador privado (só os amigos podem visualizar o seu perfil completo como jogos, equipas, resultados, amigos) ou utilizador publico (todos os membros da aplicação podem visualizar o perfil do utilizador).
Mensagens (*chat*)	Permite ao utilizador enviar e receber mensagens de qualquer utilizador da aplicação em tempo real. Os utilizadores amigos ou já contatados pela aplicação do chat entram na lista de contactos do utilizador.
Criar equipa	Permite ao utilizador criar uma equipa escolher a modalidade e localização da respetiva equipa.
Registar local	Permite ao utilizador registar um local n aplicação inserindo a posição exata, contatos do local, fotografia e o conjunto de modalidades que se pode praticar nesse local.
Pesquisar locais	Permite ao utilizador pesquisar locais com uma lista de resultados em conjunto com um mapa que indica a posição exata do local. O conjunto de locais pode ser filtrado por: **Distância:** Num determinado ponto do mapa pode restringir ou aumentar o raio de procura. **Modalidade:** Local com a modalidade que pretende realizar. **Disponibilidade:** Filtra os locais que não estão disponíveis. Neste caso um utilizador tem de estar associado a um determinado local, como proprietário do respetivo local, para ser possível gerir as reservas na aplicação. **Proprietário:** Apresenta só os locais que tenham regime de proprietário.
Visualizar local	Permite a um utilizador visualizar informação do local como contatos, fotografias, descrições, avaliação, comentários e caso for um local registado como proprietário permite visualizar a disponibilidade dos vários espaços/campos que o local poderá ter.
Avaliar um local	Permite escrever um comentário a um local e atribuir uma avaliação à localização, comodidades, materiais, infraestrutura e relação qualidade/preço.
Denunciar informação de um local	Permite denunciar informação incorreta sobre um local.
Adicionar dados em falta	Permite adicionar dados que estejam em falta num local como contactos ou fotografias.

Editar dados da página do local (Proprietário)	Permite ao proprietário do local editar todos os dados acerca do local.
Visualizar reservas (Proprietário)	Permite ao proprietário do local visualizar as reservas efetuadas pelas equipas em cada espaço/campo do local.
Gerir espaços (Proprietário)	Permite ao proprietário do local adicionar e remover espaços/campos do local.
Editar dados dos espaços (Proprietário)	Permite ao proprietário do local editar os dados do espaço como: nome, fotografia, preço, dimensões, hora de abertura e fecho, modalidades que podem ser praticadas e dias que o espaço está encerrado.
Reservar espaço (Proprietário)	Permite ao proprietário do local ocupar um espaço numa data, hora e duração.
Escrever publicações e comentários na equipa	Permite a um utilizador de uma equipa escrever publicações na equipa, deixar uma estrela (equivalente a um like facebook) numa publicação e fazer comentários sobre as publicações dessa equipa.
Visualizar jogos	Permite a um utilizador de uma equipa visualizar os jogos da equipa e juntar ao jogo caso esteja convidado.
Criar o evento	Permite a um utilizador de uma equipa com privilégios criar eventos tipo: **Treino:** Jogo entre os membros da equipa. O utilizador que cria este tipo de evento terá de escolher um nome para o evento, local, data, hora, duração e convidar os membros que pretende. Na página do treino o utilizador criador pode sempre convidar mais membros. **Duelo:** Jogo entre duas equipas. O utilizador que cira este tipo de evento terá de escolher um nome para o evento, local, data, hora, duração e a equipa adversária. O utilizador criador do evento pode alterar a equipa adversária caso demore muito tempo a responder ou caso a mesma rejeite o convite. Na página do duelo o utilizador criador pode sempre convidar mais membros para a sua equipa, bem como o capitão da equipa adversária. **Torneio:** Para a criação do torneio o capitão da equipa terá de dar um nome ao torneio, escolher o número de equipas (4, 8 ou 16), atribuir um utilizador árbitro (responsável pela atribuição dos resultados dos jogos) e enviar um convite às equipas adversárias.
Visualizar o evento treino	Permite a um utilizador de uma equipa entrar na página do treino e visualizar a informação do jogo como data, hora, duração, local, membros convidados, membros que rejeitaram o convite e os membros que vão e qual a equipa que irão alinhar. O utilizador convidado pode juntar ao jogo nesta página, alterar de equipa ou rejeitar o convite.
Visualizar o evento duelo ou torneio	Permite a um utilizador de uma equipa entrar na página do evento e visualizar a informação do jogo como data, hora, duração, local, membros convidados, membros que rejeitaram o convite e os membros que vão jogar pelas duas equipas. O utilizador convidado pode juntar ao jogo pela sua equipa.
Visualizar os resultados da equipa	Permite a um utilizador de uma equipa visualizar os resultados dos duelos e jogos dos torneios.
Inserir o resultado do duelo	Permite a um utilizador que criou o evento tipo duelo inserir o resultado bem como, o feedback da equipa adversária.

Visualizar os jogadores da equipa	Permite a um utilizador de uma equipa visualizar os jogadores pertences à equipa.
Visualizar detalhes da equipa	Permite a um utilizador de uma equipa visualizar a informação da equipa como contatos, localização, descrição, modalidade. O utilizador capitão pode alterar editar estes dados.
Convidar Jogadores para equipa	Permite a um utilizador de uma equipa com privilégios apropriados convidar um membro para juntar à equipa.
Visualizar sugestões de jogadores para a equipa	Permite a um utilizador de uma equipa com privilégios apropriados visualizar um conjunto de sugestões de jogadores para adicionar à equipa, tendo em conta a sua localização e a modalidade que o utilizador segue.
Editar privilégios	Permite a um utilizador capitão de equipa editar os privilégios de cada membro da equipa.
Visualizar torneios	Permite a um utilizador de uma equipa visualizar os torneios que está inserida a equipa ou convidada. Caso seja o utilizador capitão pode aceitar ou rejeitar o convite para o torneio.
Visualizar a página do torneio	Permite a um utilizador que tenha uma equipa no torneio visualizar os jogos dos torneios, o arbrito/responsável pelo torneio, a equipa que planeou/criou o torneio, as equipas que estão a jogar no torneio e os resultados dos jogos que já foram realizados.
Sugerir local e horário de um jogo no torneio	Permite a um utilizador capitão de equipa que está dentro de um torneio e com um jogo combinado sugerir à equipa adversária o local, data e hora do jogo.
Inserir resultado de um jogo no torneio	Permite a um utilizador da aplicação que é árbitro do torneio inserir o resultado do jogo do torneio. A equipa vencedora do jogo irá disputar a próxima fase do torneio. A equipa que saiu derrotada será eliminada do torneio.
Visualizar notificações	Permite ao utilizador visualizar as suas notificações. Será recebido uma notificação quando. -um utilizador de uma equipa faz uma publicação ou um comentário a uma determinada publicação. -um utilizador aceita um pedido de amizade. -um utilizador é adicionado à equipa que o próprio pediu convite. -uma equipa que foi convidada para um duelo aceitar ou recursar. -um utilizador tipo árbitro ou capitão da equipa tem de realizar alguma função num torneio, como sugerir o local e horário de um jogo ou em caso do árbitro para adicionar o resultado do jogo. -é efetuada uma reserva num espaço, para um evento, que no qual o utilizador é proprietário.
Visualizar convites	Permite ao utilizador visualizar os seus convites. Será recebido convites quando: -um utilizador é convidado para um evento de uma das suas equipas. -um utilizador é convidado para juntar a uma nova equipa. -um utilizador recebe um pedido de amizade. -o capitão de uma equipa recebe um convite para um duelo. -o capitão de uma equipa recebe um convite para alinhar a sua equipa num torneio. -um utilizador é convidado para ser árbitro de um torneio.
Pesquisar por localização, equipa, local ou jogador	Permite a um utilizador fazer uma pesquisa na aplicação por: Localização: uma localização geográfica, cidade, localidade que será redirecionado para o mapa da aplicação que contem os resultados dos locais na localização pretendida.

	Locais: pesquisar por um determinado local que será redirecionado para a página do local que pretende consultar. Equipas: pesquisar por equipas que será redirecionado para a página da equipa que pretende consultar. Jogador: pesquisar pelo nome de um jogador que será redirecionado para a página desse do perfil desse jogador.

Tabela 3.1 - Especificação de Requisitos

3.1.2 Requisitos não funcionais

Os requisitos não funcionais também designados de requisitos de qualidade são os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, portabilidade… Os requisitos não funcionais para este projeto são:

- **Fiabilidade –** A aplicação terá de apresentar o mínimo de falhas possíveis na sua utilização.
- **Desempenho –** A aplicação deve apresentar resultados num mínimo tempo possível.
- **Confiabilidade –** A aplicação deve ter alta disponibilidade nas suas funcionalidades.
- **Portabilidade –** A aplicação deve estar disponível em dipositivos móvel e desktop.
- **Usabilidade –** A interface da aplicação deverá ser intuitiva e fácil de usar.

3.2 Casos de Uso

Um diagrama caso de uso é a representação da interação do utilizador com a aplicação, permite identificar os vários atores presentes na aplicação e as suas funções.

Nesta aplicação foram apresentados diversos atores: utilizador, proprietário, capitão de equipa, criador eventos, convida membros para equipa e o árbitro.

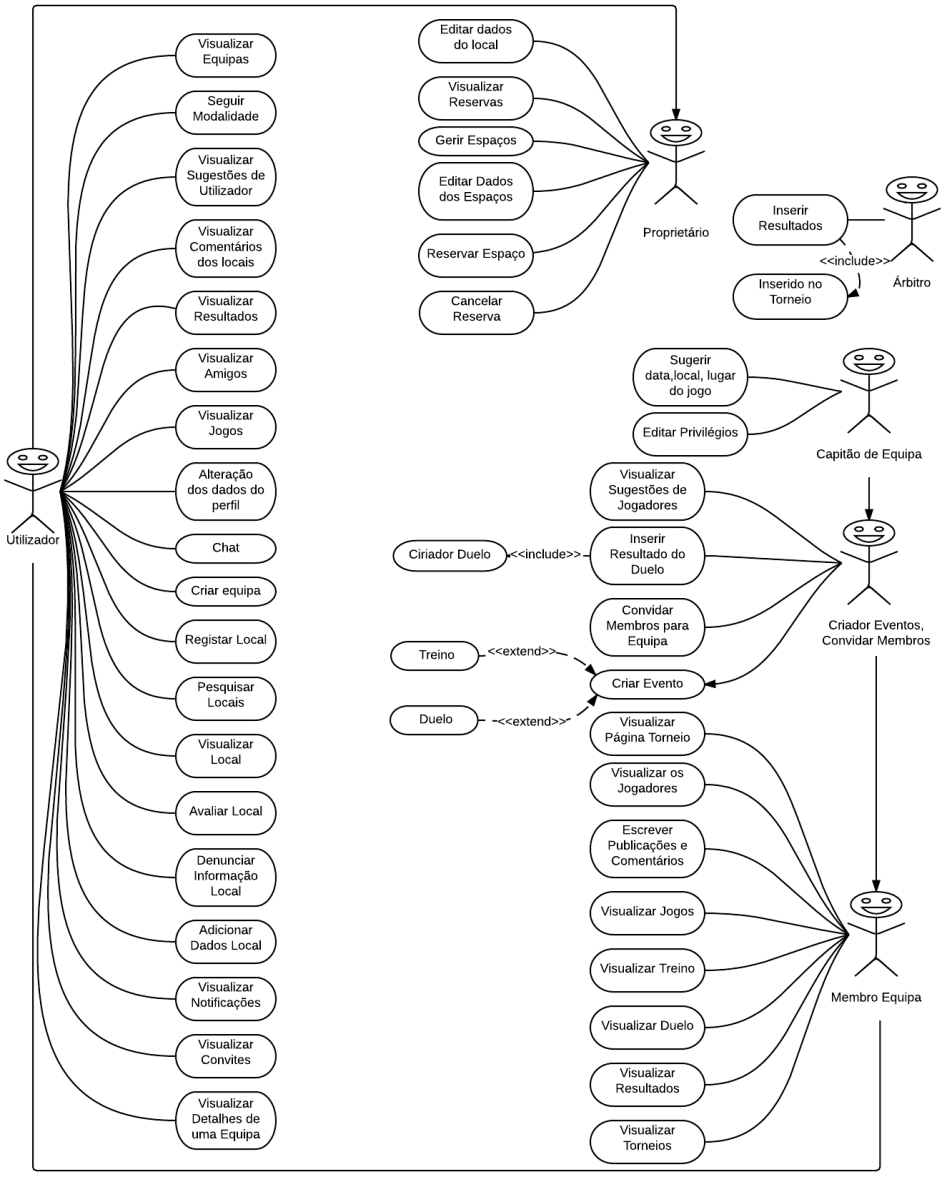

Figura 3.1 - Diagrama de Casos de Uso

Além do diagrama é necessária uma descrição narrativa de uma sequência de eventos que ocorre quando um ator usa a aplicação para realizar uma tarefa. Esta sequência de eventos é importante na implementação da aplicação pois, permite estudar a melhor forma de implementar as diversas funcionalidades. Serão apresentados 5 dos casos de uso que compõem a aplicação.

Caso de Uso	Registar Utilizador
Ator Principal	Utilizador
Pré-condições	-
Pós-condições	É criado um novo registo de utilizador. O utilizador é adicionado ao catálogo de utilizadores com as diversas modalidades que pretende seguir.

1. O utilizador configura o perfil com o nome de utilizador e *password*.
2. A aplicação verifica se o nome de utilizador não existe e se a *password* está conforme os critérios.
3. O utilizador repete o passo 1 até receber confirmação do sucesso do registo.
4. A aplicação apresenta um conjunto de modalidades ao utilizador
5. O utilizador escolhe a modalidade que pretende seguir.
6. A aplicação regista a modalidade escolhida
7. O utilizador repete os passos 5 a 6 até estar satisfeito com as modalidades a seguir.
8. O utilizador confirma que terminou a inserção de modalidades
9. A aplicação cria um novo id para o utilizador e adiciona-o ao catálogo de utilizadores

Tabela 3.2 - Caso de Uso Registar Utilizador

Caso de Uso	Registar local
Ator Principal	Utilizador
Pré-condições	O utilizador está registado na aplicação
Pós-condições	A aplicação regista o local e adiciona ao catálogo de locais

1. O utilizador indica a localização e os dados do local
2. A aplicação verifica se o local já se encontra e regista os dados.
3. A aplicação apresenta o conjunto de modalidades.
4. O utilizador indica a modalidade
5. A aplicação regista a modalidade
6. O utilizador repete os passos 4 a 5 até indicar que terminou.
7. A aplicação regista o local.

Tabela 3.3 - Caso de Uso Registar Local

Caso de Uso	Avaliar Local
Ator Principal	Utilizador
Pré-condições	O utilizador está registado na aplicação e nunca avaliou o local
Pós-condições	A aplicação regista o comentário e a pontuação e atualiza a pontuação

1. O utilizador escreve o seu comentário.
2. A aplicação regista o comentário.
3. A aplicação aplica a pontuação de cada categoria (localização, comodidades, materiais, infraestrutura, relação qualidade/preço)
4. A aplicação regista a pontuação por categoria.
5. A aplicação realiza a media da pontuação obtida e atualiza a pontuação total.

Tabela 3.4 - Caso de Uso Avaliar Local

Caso de Uso	Criar Equipa
Ator Principal	Utilizador
Intervenientes	Utilizador e utilizadores que estão a um raio de km desejado e que seguem uma determinada modalidade
Pré-condições	O utilizador está registado na aplicação
Pós-condições	A aplicação regista a equipa adicionando ao catálogo de equipas e todos os respetivos membros que convidou para aderir à equipa.

1. O utilizador indica o nome da equipa.
2. A aplicação verifica se o nome não existe.
3. O utilizador repete o passo 1 até receber confirmação do sucesso do nome da equipa.
4. A aplicação pergunta o qual a modalidade que a equipa vai disputar.
5. O utilizador indica a modalidade
6. A aplicação regista a modalidade.
7. O utilizador indica a localização da equipa
8. A aplicação regista a localização
9. O utilizador indica que pretende adicionar membros à equipa
10. A aplicação filtra os membros de acordo com a sua localização.
11. O utilizador escolhe o membro para adicionar à equipa.
12. A aplicação envia um pedido ao utilizador convidado.
13. O utilizador repete os passos 11 a 12 até estar satisfeito com o número de membros que pretende convidar.
14. O utilizador indica que já terminou os envios de pedidos.
15. A aplicação regista a equipa.

Tabela 3.5 - Caso de Uso Criar Equipa

Caso de Uso	Criar Evento
Ator Principal	Utilizador
Intervenientes	Utilizadores da equipa
Pré-condições	O utilizador está registado na aplicação, está dentro da página da equipa e tem permissões para criar o evento
Pós-condições	A aplicação regista o evento e adiciona ao catálogo de eventos. A aplicação regista o local por disponibilidade

1. O utilizador escolhe um tipo de evento.
2. O utilizador indica o nome do evento.
3. A aplicação regista o nome do evento.
4. A aplicação pergunta o dia e as horas que pretende realizar o jogo.
5. O utilizador indica as horas e o dia.
6. A aplicação regista as horas e o dia.
7. O utilizador inicia a convocatória.
8. A aplicação indica os jogadores que pertencem à equipa.
9. O utilizador convida o membro.
10. A aplicação envia o convite ao membro.
11. O utilizador repete os passos 9 a 10 até estar satisfeito com o número de membros que pretende convidar.
12. O utilizador indica que já terminou os envios de pedidos.
13. A aplicação apresenta conjunto de locais disponíveis.
14. O utilizador escolhe o local.
15. A aplicação apresenta os vários espaços/campos disponíveis no local.
16. O utilizador escolhe o espaço.
17. A aplicação reserva o espaço e envia a informação da reserva ao proprietário.

Tabela 3.6 - Tabela Criar Evento

Caso de Uso	Registar Resultado do Duelo
Ator Principal	Utilizador
Pré-condições	O utilizador está registado na aplicação e é capitão de equipa
Pós-condições	A aplicação regista o resultado e notifica o resultado ao adversário

1. O utilizador indica o resultado do duelo.
2. A aplicação regista o resultado.
3. O utilizador indica o feedback da equipa adversária
4. A aplicação regista o feedback e atualiza o feedback total da equipa adversária
5. A aplicação notifica o capitão da equipa adversária.

Tabela 3.7 - Criar Registar Resultado do Duelo

3.3 Modelo de Domínio

O modelo de domínio é um diagrama que descreve o comportamento da aplicação. O diagrama descreve um conjunto de conceitos significativos do mundo real para que possa ser modelado mais tarde na aplicação. Estes conceitos incluem regras de negócio utilizando uma relação com os dados.

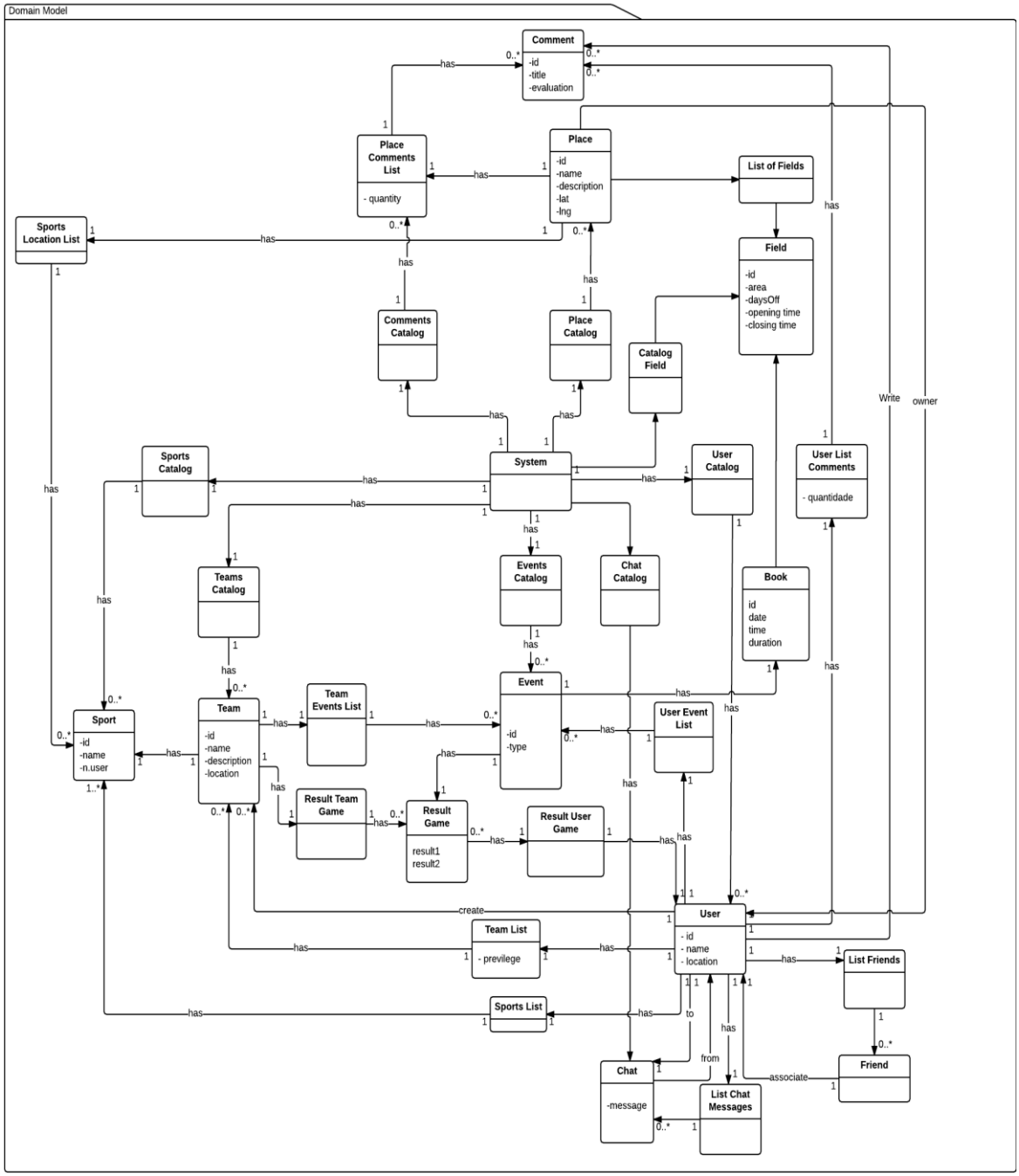

Figura 3.2 - Modelo de Domínio

3.4 Esboços da aplicação web

Os esboços foram essenciais no final da análise de requisitos para obter uma melhor abordagem sobre a aplicação web. Através dos esboços foi possível estudar as interfaces que iriam constituir a aplicação e em que interfaces seriam localizados os requisitos referido na secção 3.1.1 . As interfaces das redes sociais famosas como é o caso do Facebook[11], Twitter[12], Instagram[13] e Google Plus[14] os utilizadores já se encontram bastantes familiarizados, sendo que entre elas tem muitos aspetos em comuns como por exemplo as notificações e os convites serem uma janela que se abre no cabeçalho da página, as publicações e os cometários são funcionam praticamente do mesmo gênero em todas as redes sociais. Todos estes aspetos foram importantes e devem permanecer na rede social, de forma a que os utilizadores conseguissem interagir com a aplicação com a mesma facilidade das aplicações famosas referidas. De seguida estão representados alguns dos esboços fundamentais para a construção da aplicação.

Na Figura 3.3 é apresentado o esboço da página inicial cujo o objetivo é o login e registo de um novo utilizador. Na fase de registo existem poucos campos para incentivar o utilizador a realizar o registo.

Figura 3.3 - Página inicial de login/registo

[11]www.facebook.com

[12]www.twitter.com

[13]www.instagram.com

[14]www.plus.google.com

Na Figura 3.4 é apresentado o esboço da página inicial do utilizador tem toda a sua informação relativa à sua conta.

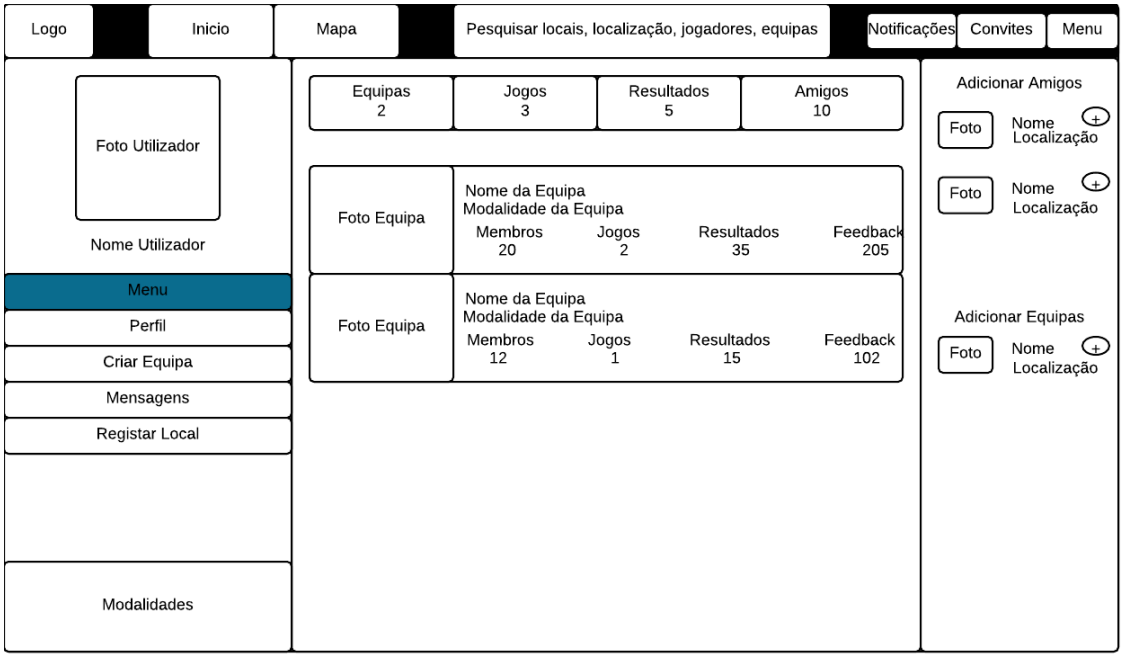

Figura 3.4 - Pagina inicial do utilizador

Na Figura 3.5 é apresentado um esboço onde um utilizador realizará a pesquisa de locais, podendo filtrar os resultados apresentados. Os resultados serão apresentados em conjunto com um mapa.

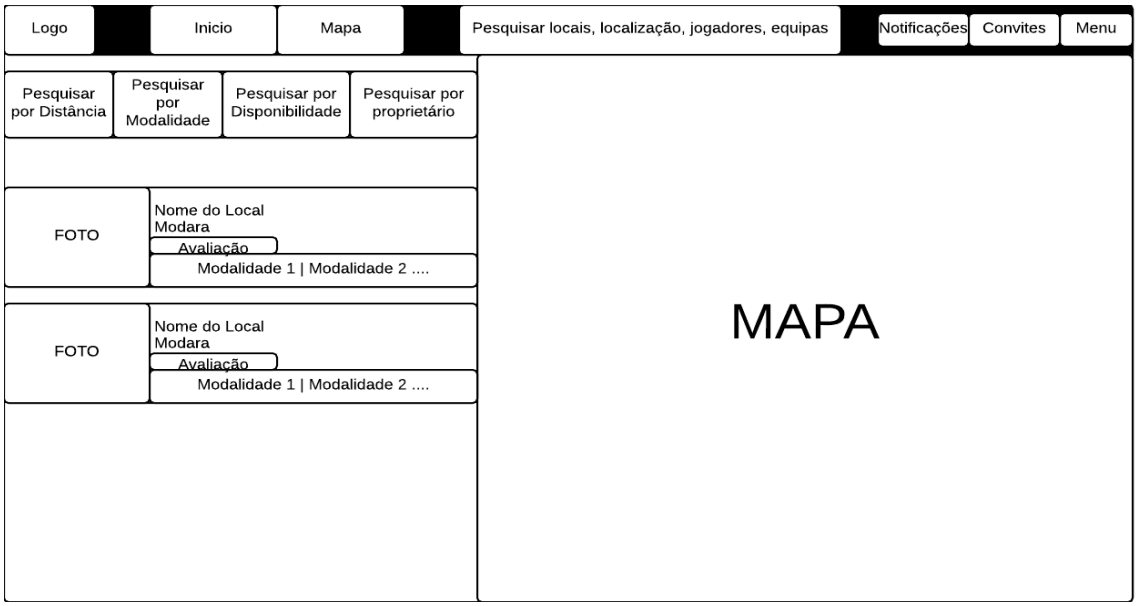

Figura 3.5 - Página de procura de locais por localização

Na Figura 3.6 é apresentado o esboço da página do local onde o utilizador poderá visualizar informação importante.

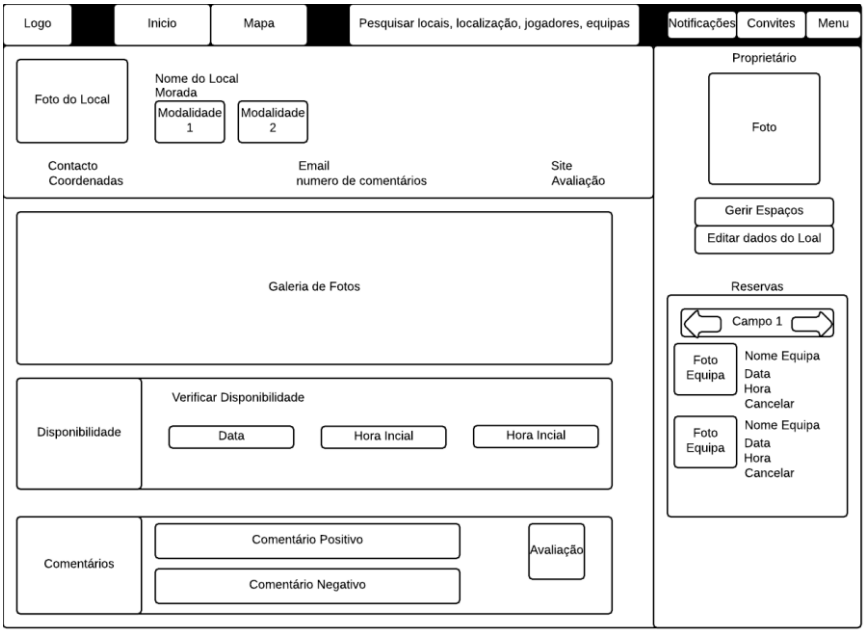

Figura 3.6 - Página do Local com proprietário autenticado

Na Figura 3.7 é apresentado o esboço de um duelo entre duas equipas em que se pode visualizar os convidados e interessados no duelo.

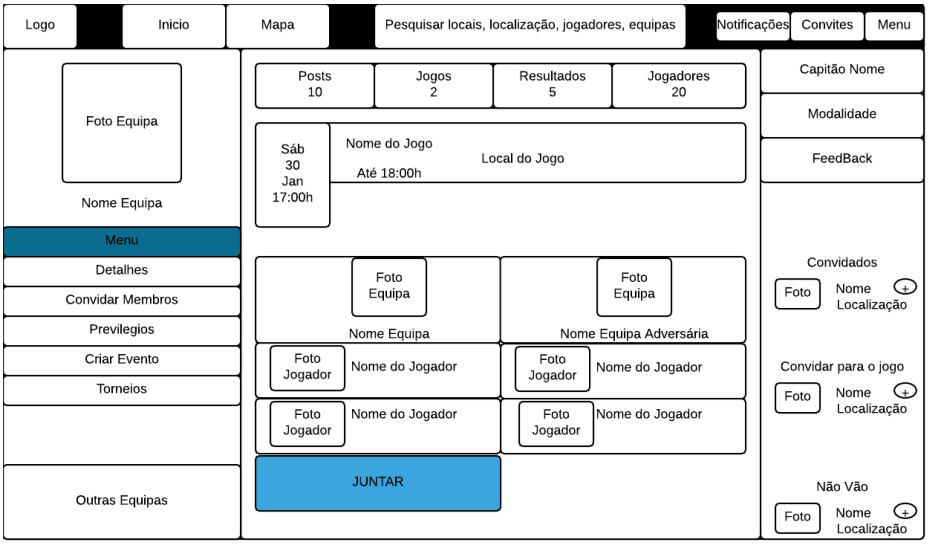

Figura 3.7 - Página do Duelo na Equipa

3.5 Recolha de dados de infraestruturas desportivas

A recolha de dados foi realizada durante a analise de requisitos até ao início do desenvolvimento da aplicação. A recolha de dados consistiu em encontrar informação importante como nome do local, fotografias, contatos, descrições, localização em coordenadas e modalidades que se podia realizar. No total foram encontrados 804 locais, representado quase na totalidade o número de infraestruturas desportivas disponíveis em todo o país, sendo que na totalidade dos locais recolhidos seja possível realizar 1670 modalidades. Com a quantidade de informação de infraestruturas recolhidas, um utilizador em qualquer ponto do país pode facilmente encontrar um local para realizar o seu deporto perto da sua residência.

3.6 Plano de Trabalho

Nesta secção é apresentada a calendarização do projeto, bem como a explicação das alterações que foram feitas em relação ao plano inicial proposto.

3.6.1 Calendarização do Projeto

Este projeto foi organizado de acordo com as seguintes fazes do ciclo de vida:

3.6.2 Planeamento do projeto de 15 a 30 Setembro

- Integração na equipa do projeto
- Planeamento do projeto

3.6.3 Análise de Requisitos de 1 a 10 de Outubro

- Descrição da aplicação
- Requisitos funcionais
- Requisitos não funcionais
- Diagramas de casos de uso
- Esboços de interfaces

3.6.4 Desenho de Software de 11 de Outubro a 21 Outubro

- Modelo de Domínio
- Sistema de Diagramas de Sequência (SSDs)
- Modelo de Dados

3.6.5 Desenvolvimento de 22 Outubro a 25 de Abril

- Aquisição dos dados dos locais

    - Contactar associações desportivas (22 de Outubro)
    - Inserção dos dados na base de dados (22 de Outubro a 1 de Janeiro)
- Back End
    - Construção da base de dados (22 a 23 Outubro)
    - Sistema de registo e autenticação pela aplicação (24 a 27 de Outubro)
    - Configuração de perfil do utilizador (28 a 31 de Outubro)
    - Sistema de pesquisa de locais (1 a 15 de Novembro)
    - Desenvolvimento da página do local (16 a 18 Novembro)
    - Sistema de avaliações e comentários de locais (19 a 25 Novembro)
    - Página de utilizador (26 Novembro a 12 Dezembro)
    - Sistema de pesquisa por localização, locais, equipas e jogadores (13 Dezembro a 18 Dezembro)
    - Sistema de equipas (19 Dezembro a 27 Janeiro)
        - Sistema de convite de membros
        - Sistema de privilégios
        - Sistema de eventos (Treinos e Duelos)
        - Sistema de resultados
        - Sistema de atualização de dados
    - Sistema de proprietários de locais (22 Janeiro a 15 Fevereiro)
        - Sistema de disponibilidade
        - Sistema de reservas
        - Criação de espaços/campos
        - Sistema de consulta de reservas
    - Sistema de mensagens (chat) (16 Fevereiro a 27 Fevereiro)
    - Desenvolvimento dos eventos tipo torneios (28 Fevereiro a 16 Março)
    - Sistema de autenticação/registo pelo Facebook (16 a 19 Março)
- Front-end
    - Desenvolvimento das *views* para *desktop* (20 Março a 9 Abril)
    - Adaptação das *views* a dispositivos móveis (10 Abril a 30 Abril)
- Relatório Final (1 a 30 de Maio)
- Testes de usabilidade (15 de Maio a 30 Maio)

3.7 Alteração no planeamento do projeto

A alteração do projeto começou logo no início do desenvolvimento quando foi encontrado uma aplicação muito semelhante à que era proposto fazer, o que acabou por aumentar de forma significativa o número de funcionalidades. Primeiro foi decidido que a aplicação seria mais prestigiosa se fosse uma rede social desportiva. Para que a aplicação se tornasse uma rede social teria de ser possível que os utilizadores tivessem uma maior interação. Desta forma, foram adicionadas várias funcionalidades que permitiram concretizar uma rede social desportiva, como por exemplo um sistema em que os utilizadores conseguissem adicionar-se como amigos, de forma a conseguissem comunicar por um sistema de *chat* em tempo real. Também mais tarde pensou-se num problema que iria surgir por parte dos utilizadores. De facto, a aplicação permitia uma equipa escolhesse um local para jogar, mas esse local poderia estar ocupado, o que seria um problema para essa equipa, então surgiu a ideia de criar um sistema de reservas de locais. Então um utilizador poderia pesquisar por um local que estivesse disponível num certo dia e hora e numa determinada duração. Para além deste sistema de reserva também seria necessário ser criado um sistema que permitisse um utilizador dono/proprietário do local controlar as suas reservas e inserir os vários espaços/campos que local tivesse e o seu horário e dias de encerramento. E por fim foi realizado mais um tipo de evento, o torneio, em que permite que um conjunto de equipas se defronte nos locais e datas que as equipas pretenderem sendo que esse local e datas combinadas e aceites por ambas as equipas. Mas surgia um problema pois, poderia haver uma manipulação de resultados para isso, no torneio é convidado um árbitro que irá ser responsável por inserir os resultados dos jogos.

Capítulo 4

Implementação da aplicação web

4.1 Arquitetura da Aplicação

A programação por camadas é uma técnica muito comum para controlar a complexidade de um sistema. É possível especializar-se numa camada sem saber muito mais sobre o sistema. Esta arquitetura por camadas irá minimizar a dependência entre camadas, cada camada interage com as camadas vizinhas e não necessita de saber detalhes das restantes além disso cada camada construída fornece novos serviços de alto nível a camadas futuras sendo que o número excessivo de camadas pode afetar o desempenho, dado que entre cada interface é necessário realizar transformações na informação transmitida. É necessário tomar a decisão de quais são as camadas a implementar e quais as suas responsabilidades. Na Figura 4.1 é apresentado a arquitetura da aplicação, que é constituída por três camadas: camada de apresentação, camada de lógica e camada de dados. Esta secção foi baseada na seguinte referencia [6].

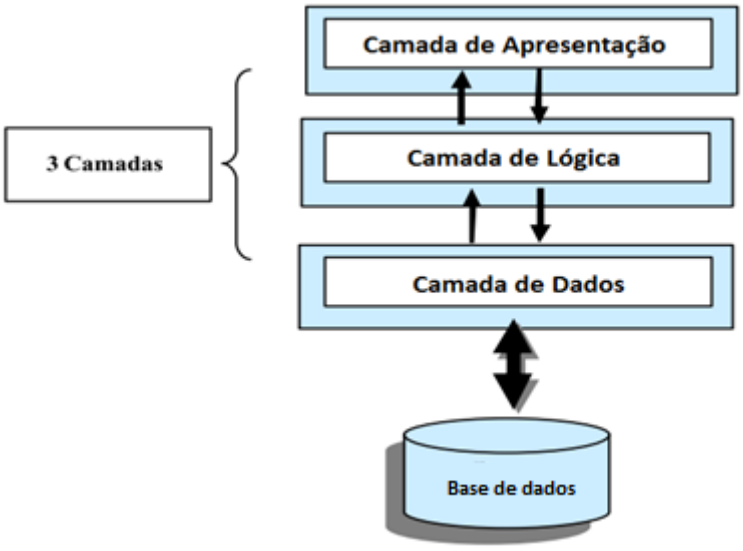

Figura 4.1 - Arquitetura da aplicação por camadas

Camada de Apresentação: tem como responsabilidade a interação entre utilizador e a aplicação que no qual é apresentada a interface.

Camada de Lógica ou Negocio: engloba todo o trabalho relacionado com o domínio onde a aplicação se encontra (regras de negocio, funcionalidades). Esta camada é a camada que faz de intermediário entre a camada de apresentação e a camada de dados, ou seja, quando um utilizador acede aos dados na camada de dados é através de um pedido realizado da camada de apresentação, no entanto este pedido não é realizado diretamente tendo com o intermediário a camada de lógica.

Camada de Dados: tem como objetivo de controlar a comunicação com aplicações terceiras, em especial gerir a base de dados relacional que armazena a informação persistente da aplicação.

4.2 Estrutura da aplicação

A *freamework codeigniter* cria uma estrutura de pastas para facilitar a organização do código. Na Figura 4.2 está representada a estrutura de páginas da aplicação.

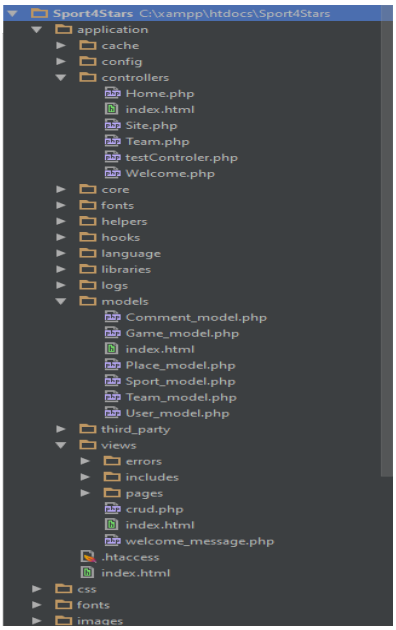

Figura 4.2 - Estrutura de páginas da aplicação

Pasta da aplicação – Esta pasta contem os documentos que vão ser responsáveis pela criação da aplicação. Os documentos que irão fazer parte desenvolvimento da aplicação estão nas pastas models, as views e controllers.

Pasta cache – A cache contém todas as páginas em cache da aplicação. Irá servir sobre tudo para a aplicação ter um melhor desempenho.

Pasta config – Esta pasta permite definir as configurações da aplicação como o *url* base, a página de índex, a configuração para a base de dados etc…

```
$active_group = 'default';
$query_builder = TRUE;

$db['default'] = array(
    'dsn'   => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'olympicsports',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => (ENVIRONMENT !== 'production'),
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);
```

Figura 4.3 - Configuração da base de dados

Pasta controladores – Esta pasta irá conter os arquivos classe desenvolvida para a aplicação. O *codeigniter* por defeito vem com um *controller* chamado *Welcome,* este é o *controller* inicial que é chamado cada vez que seja executada a aplicação, podendo ser substituído mais tarde no ficheiro *config.php* na pasta *config.*

```
class Welcome extend CI_Controller {
```

Figura 4.4 - *Controller* inicio da *class*

Class – A class irá definir o tipo de item neste caso, quando é aberto o *browser* e digitado o link www.exemplo.com/welcome/index ou neste caso como é o *index* simplesmente, www.exemplo.com/welcome/ vai obter as funcionalidade do controlador *Welcome* no método índex pois, será o método inicial do controlador. O *extends* CI_Controller o controlador irá conter todas as funcionalidades principais do controlador ou seja, vai herdar a capacidade de usar as funcionalidades.

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Welcome extends CI_Controller {

	/**
	 * Index Page for this controller.
	 *
	 * Maps to the following URL
	 * 		http://example.com/index.php/welcome
	 *	- or -
	 * 		http://example.com/index.php/welcome/index
	 *	- or -
	 * Since this controller is set as the default controller in
	 * config/routes.php, it's displayed at http://example.com/
	 *
	 * So any other public methods not prefixed with an underscore will
	 * map to /index.php/welcome/<method_name>
	 * @see http://codeigniter.com/user_guide/general/urls.html
	 */
	public function index()
	{
		$this->load->view('welcome_message');
	}
}

/* End of file welcome.php */
/* Location: ./application/controllers/welcome.php */
```

Figura 4.5 - *Controller* inicial do *Codeigniter*

Core – Nesta pasta serão encontrados os arquivos bases que pertencem à aplicação. Quando a aplicação do *codeigniter* correr as classes base são inicializadas automaticamente.

Erros – Nesta pasta é específica para os *logs* de erros.

Helpers – Esta pasta contem um conjunto de funções para ajudar os programadores a realizarem uma determinada tarefa. Este conjunto de *helpers* são escritos em formato Orientado a Objetos. Cada função *helper* executa uma tarefa especifica, sem qualquer dependência com outras funções. Exemplos de helpers podem ser URL, formulário, *cookies*, arquivo. Na Figura 4.6 é apresentado como é carregado, neste caso o *helper* é carregado no construtor do controlador, ou seja o *helper* poderá ser utilizado em todo o controlador, se fosse apenas necessário ser usado num método poderia ser carregado apenas nesse método, caso seja para utilizar na aplicação toda poderá ser carregado no ficheiro autoload.php na pasta config Figura 4.7.

```php
public function __construct()
{
    parent::__construct();
    $this->load->helper('url');
    $this->load->helper('form');
    $this->load->helper('array');
}
```

Figura 4.6 - Carregamento do *Helper* no construtor do *Controller*

```php
/*
| -------------------------------------------------------------------
|  Auto-load Helper Files
| -------------------------------------------------------------------
| Prototype:
|
|   $autoload['helper'] = array('url', 'file');
*/
$autoload['helper'] = array('url','file','array');
```

Figura 4.7 – Carregamento do *Helper* na aplicação

Na Figura 4.8 pode ser visualizado a construção de um formulário com a chamado do *helper "form"*, com poucas linhas de código é possível substituir todo o HTML. No entanto, quando a página é executada todo este código será traduzido para o HTML.

```php
<div id="container">
    <?php echo form_open('main_controller'); ?>
    <h1>Create Contact Form Using CodeIgniter</h1>
    <?php echo form_label('Student Name :'); ?>
    <?php echo form_input(array('id' => 'dname', 'name' => 'dname')); ?>
    <?php echo form_label('Student Email :'); ?>
    <?php echo form_input(array('id' => 'demail', 'name' => 'demail')); ?>
    <?php echo form_label('Student Mobile No. :'); ?>
    <?php echo form_input(array('id' => 'dmobile', 'name' => 'dmobile')); ?>
    <?php echo form_label('Student Address :'); ?>
    <?php echo form_input(array('id' => 'daddress', 'name' => 'daddress')); ?>
    <?php echo form_submit(array('id' => 'submit', 'value' => 'Submit')); ?>
    <?php echo form_close(); ?>
</div>
```

Figura 4.8 - Criação de um Formulário usado *helper Form*

Language – Nesta pasta será encontrado um conjunto de ficheiro de línguas, sendo que poderá ser criada ficheiros pelos próprios programadores para alterar a linguagem de erros ou outras mensagens importantes na linguagem que se pretenda. Na Figura 4.9 mostra um exemplo como se carrega o ficheiro language.php e na Figura 4.10 um exemplo de aplicação.

```
$this->lang->load('filename', 'language');
```

Figura 4.9 - Carregamento do ficheiro *language*.

Esta linha de código na Figura 4.10 deverá estar no ficheiro *config.php* na pasta *config*.

```
$lang['error_email_missing'] = "You must submit an email address";
$lang['error_url_missing'] = "You must submit a URL";
$lang['error_username_missing'] = "You must submit a username";
```

Figura 4.10 - Exemplo de utilização do sistema *language*

Libraries – Nesta pasta são colocadas as *libraries* desenvolvidas pelo programador que serão uteis para a aplicação. *$this*, no entanto, só trabalha diretamente dentro dos *controllers*, *models*, ou as *views*. Também é possível usar classes do *codeigniter* dentro das próprias classes personalizadas. As *libraries* são carregadas de igual forma como os *helpers*.

```
$this->load->library('form_validation');
```

Figura 4.11 - Carregamento da *library*

Na Figura 4.12 é apresentado um exemplo de verificação de um formulário. É de notar que graças a este método não necessitamos de realizar qualquer código *javascript* para fazer a validação do respetivo formulário poupando imenso código, sendo possível editar as mensagens de erro para cada uma das regras aplicadas.

```php
public function create() {
    $this->form_validation->set_rules('lat', 'Localizacao', 'trim|required');
    $this->form_validation->set_rules('name', 'Nome', 'trim|required|is_unique[team.name]|max_length[20]');
    $this->form_validation->set_rules('checkbox_sport', 'Desporto', 'required');
    $this->form_validation->set_message('required', '%s é obrigatório');
    $this->form_validation->set_message('is_unique[team.name]', 'O campo Nome da Equipa já existe');
    $this->form_validation->set_message('max_length[20]', 'O campo nome não pode conter mais de 20 caracteres');
    if ($this->form_validation->run() == TRUE) {
        $this->load->view('form_page');
    }
    $this->load->view('form_success');
}
```

Figura 4.12 - Validação de um *form* utilizado a *library form_validation*

Em cada campo pode ser adicionado várias regras. No primeiro parâmetro da função *set_rules* é designado o nome do campo, no segundo é a designação do nome do campo ao utilizador, que no qual em caso de erro será mostrado para diferenciar dos outros campos, o terceiro parâmetro são as regras que vão ser atribuídas ao campo, como por exemplo "*is_unique[table_name.colum_name]*" que significa que é único na base de dados numa dada tabela num dado campo, *required* este campo é obrigatório. Em caso de erro poderá ser mostrada uma mensagem que o programador pretender.

Models: esta pasta será a que irá conter os *models* do projeto. Estes *models* são as classes php que são projetados para trabalhar com informações na base de dados. Mais informação sobre os *models* poderá ser encontrado na secção 2.1

Views: A pasta views vão ser onde irá ficar as paginas web. Estas *views* nunca são chamadas diretamente pois devem ser carregadas a partir do *controller*. Como foi referido na secção 2.1 o *codeigniter* utiliza o padrão de arquitetura de *software* MVC. Na Figura 4.13 mostra como é carregado a *view* no controlador.

```php
class Home extends CI_Controller
{

    public function index()
    {
        $this->load->view('index_page');
    }

}
```

Figura 4.13 - Carregamento da *view* no controlador

Quando o *controller* chama uma *view* existe uma *view* principal que é executada. Isto irá possibilitar uma melhor organização e irá evitar a duplicação de código, pois as paginas web da aplicação terão o mesmo *header* e *footer*. Na Figura 4.14 é apresentada o conteúdo do ficheiro que terá a *view* inicial e na Figura 4.15 como é chamado.

```php
<?php
    $this->load->view('includes/header');
    if($type_page === 'include_logged') {
        $this->load->view('includes/header_bar');
        if($this->uri->segment(1) === 'site' || $page === 'createTeam') {
            $this->load->view('includes/user_left_top_bar');
        }else if($this->uri->segment(1) === 'team'){
            $this->load->view('includes/team_left_bar');
        }
        if($page === 'createField' || $page === 'result') {
            $this->load->view('includes/place_left_top_bar');
        }
        if($page!='') { $this->load->view('pages/'.$page); }
        if(($this->uri->segment(1) === 'site' || $page === 'createTeam')) {
            $this->load->view('includes/user_right_footer_bar');
        }else if($this->uri->segment(1) === 'team' && $page !== 'tormentPage') {
            $this->load->view('includes/team_right_footer_bar');
        }
        if($page === 'createField' || $page === 'result') {
            $this->load->view('includes/place_right_footer_bar');
        }
    }else {
        if($page!='') { $this->load->view('pages/'.$page); }
    }
    $this->load->view('includes/footer');
```

Figura 4.14 - *View* inicial da aplicação

```php
public function index()
{
    $this->regist_user();
    $data = array(
        'title' => 'Registo',
        'page' => 'register',
    );
    $this->load->view('crud', $data);
}
```

Figura 4.15 - Chamada de uma *view* por parte do controller

4.3 Views

Cada *view* que vai ser apresentada irá ter diversas funcionalidades que compõem a aplicação web.

4.3.1 *View* Login

Esta *view* é responsável pelo registo e autenticação dos utilizadores. O registo e autenticação poderá ser realizada pela aplicação ou pelo o Facebook.

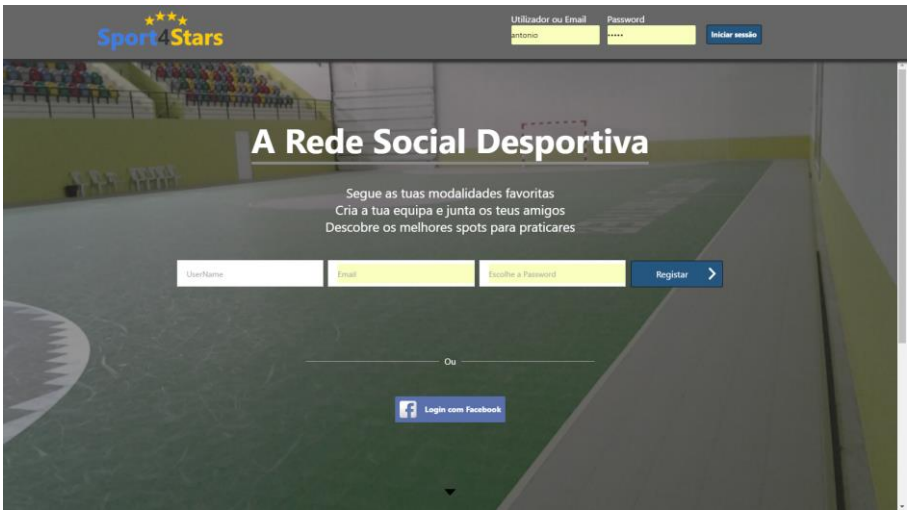

Figura 4.16 - *View* de autenticação

No registo do sistema da aplicação o utilizador terá de escolher um *UserName*, o *email* e uma password e clicar no botão registar. Estes dados vão ficar inseridos na base dados e o utilizador irá ser redirecionado para uma *view* seguinte para completar os dados do seu perfil. No registo do Facebook o utilizador só terá de clicar do botão "Login com Facebook" e será redirecionado para a mesma *view* do registo do sistema onde terá de completar os dados do seu perfil.

Para ser possível autenticar pelo sistema do Facebook foi necessário registar uma nova *app* com uma conta de Facebook e preencher os dados como o nome da aplicação o *url* etc… O próximo passo foi fazer *download* do ficheiro Facebook SDK, que contem um conjunto de API's, inserir os ficheiros nas *libraries* do *codeigniter* e chamar a *library* no controlador onde se encontra a função de login na Figura 4.17.

```
public function __construct()
{
    parent::__construct();
    $this->load->library('facebook',
        array(
            "appId"=>'144302155965976',
        "secret"=>'**************************'));
}
```

Figura 4.17 - Chamada da *library* do Facebook

Como se pode visualizar na Figura 4.17 o *array* que contem a chamada da *library* tem dois parâmetros “appId” e “secret” estes parâmetros são os que referenciam a aplicação registada que no qual pode ser obtido na pagina da aplicação no Facebook.

A ideia do registo pelo Facebook é obter os dados a fim de registar na nossa base de dados na Figura 4.18 é mostrado o código responsável por obter esses dados de uma conta do Facebook.

```
public function loginFacebook(){
    if($this->user) {
        try{
            $user_profile = $this->facebook->api('/me');
            print_r($user_profile);

        }catch (FacebookApiException $e) {
            print_r(e);
            $user = null;
        }
    }
    if($this->user) {
        $logout = $this->facebook->getLogoutUrl(array("next"=>base_url().'login/logout'));
        echo '<a href="'.$logout.'"</a>';
    }else {
        $login = $this->facebook->getLoginUrl(array("scope"=>'email'));
        echo '<a href="'.$login.'"</a>';
    }
}
```

Figura 4.18 - Código responsável por obter dados de uma conta do Facebook

Na variável user_profile é responsável por armazenar os dados do Facebook. Se tiver apenas ‘/me’ será armazenado apenas o nome do utilizador e o respetivo id. Caso seja necessário mais dados do utilizador terá de acrescentar informação a essa linha de código.

```
$user_profile = $this->facebook->api('/me?fields=name,email,first_name,last_name');
```

Figura 4.19 - Código para obter nome, email, primeiro e ultimo nome do Facebook

Uma conta de Facebook que é realizado o login pela primeira vez na aplicação será redirecionada para a página de perfil para completar os restantes dados que são necessários. Quando é realizado o próximo login o utilizador já não deve ser redirecionado para esta *view* mas sim, para a *view* inicial do seu utilizador. Para verificar se uma conta de Facebook já está registada na aplicação é guardado o seu id de Facebook na base de dados. Assim, cada vez que um utilizador realizar um login na aplicação, a aplicação será responsável por verificar se a conta de Facebook já está registada.

4.3.2 *View* Inicial

Esta é a *view* que é apresentada depois de ser realizado o login.

Figura 4.20 - Barra *header* do utilizador

A barra apresentada na Figura 4.20 é mostrada em todas as *views* em que o utilizador esteja autenticado. É composta por dois botões de redireccionamento para a página inicial do utilizador e para o mapa. Depois é apresentada uma caixa de texto onde é possível pesquisar por localização, locais, equipas e utilizadores de toda a aplicação Figura 4.21.

Figura 4.21 - Campo de pesquisa

Campo de pesquisa permite dois tipos de pesquisa, o por categoria e normal. A pesquisa por categoria é apresentada apenas os resultados da categoria Figura 4.22.

Figura 4.22 - Pesquisa por categoria de Jogadores

De início são apresentados 5 resultados caso ainda não tenha encontrado o jogador que pretende pode ir digitando teclas para filtrar o nome pretendido. No caso de existirem mais de 5 registos com o mesmo nome pode clicar “mostrar mais” em que será apresentado novas previsões. Os resultados em todas as categorias são ordenados pela localização dos utilizadores.

Para ativar a pesquisa normal é apenas necessário começar a digitar um nome sem

escolher uma categoria. O resultado é a união de todas as previsões das categorias como mostra a Figura 4.23.

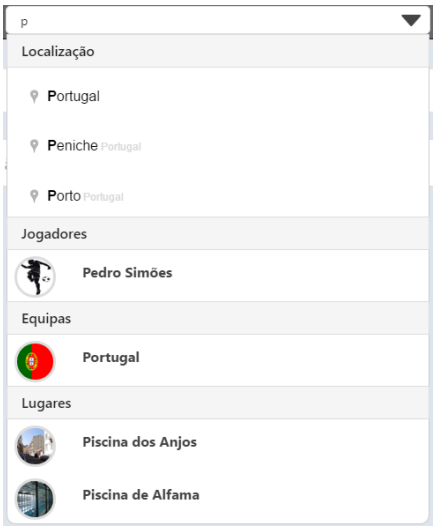

Figura 4.23 - Pesquisa normal

Para obter dados na categoria localização é utilizado a API v3 do Google Maps que utiliza um serviço *Geolocation* fornecido pelo HTML *Geolocation* API. A Figura 4.24 apresenta o código responsável por apresentar o autocomplete. A variável *input* está preparada para receber o texto que o utilizador introduzir quando é introduzido um texto o Google Maps vai criar uma *div* no HTML que irá conter a previsão dos resultados. Quando o utilizador selecionar um local o "if(!place.geomety)" irá verificar se esse local é válido, caso for válido irá chamar a função "codeAddress" que irá traduzir o endereço do local em coordenadas e redirecionar a pagina para a *view* do mapa.

```
var input = /** @type {!HTMLInputElement} */ (
    document.getElementById('autocomplete'));
var autocomplete = new google.maps.places.Autocomplete(input);
autocomplete.addListener('place_changed', function () {
    var place = autocomplete.getPlace();
    if (!place.geometry) {
        window.alert("O local tem de estar dentro da lista");
        return;
    } else {
        codeAddress($('#autocomplete').val());
    }
});

function codeAddress(address) {
    var geocoder = new google.maps.Geocoder();
    geocoder.geocode({
        'address': address
    }, function (results, status) {
        var NewMapCenter = results[0].geometry.location;
        var lat = NewMapCenter.lat();
        var lng = NewMapCenter.lng();
        var location = $('#autocomplete').val();//.replace(/#|_|,/g,'_').replace(/\(|\)/g, "");
        window.location.href = "<?php echo base_url().'site/list_result?local='?>" + location +
            "&lat=" + lat + "&lng=" + lng;
    });
}
```

Figura 4.24 - Código *Autocomplete* Google Maps

Nesta *view* inicial é também mostrado um painel do lado direito onde é apresentado um conjunto de sugestões de amigos como mostra a Figura 4.25. Para enviar o convite de amizade, ou seja, aceitar a sugestão é apenas necessário clicar no botão “mais”. Caso o utilizador pretenda visualizar mais sugestões poderá faze-lo clicando no botão “mostrar mais”. As sugestões são ordenadas de acordo com as modalidades seguidas pelos utilizadores e pela localização. Nesta página inicial além das sugestões de amizade também poderá haver sugestões de equipas.

Figura 4.25 - Sugestões de Amigos

O sistema de sugestões foi baseado na tecnologia AJAX, permitindo que cada vez um utilizador enviasse um pedido de amizade a pagina não tivesse que fazer um *reload*.

Na Figura 4.26 é apresentado o código em AJAX que é executado quando o utilizador envia um pedido de amizade. Os parâmetros que o AJAX recebe como o *url* referencia um ficheiro que está no *controller site* no método *addFriend*, o parâmetro *data* contem o *idUser*, que irá ser enviado também para o método no *controller* que indica o *id* do utilizador que queremos adicionar. Em caso de sucesso a *div* correspondente será removida do HTML isto, tudo sem realizar qualquer *refresh* na página.

```
function addFriend(){
    $.ajax({
        type: "POST",
        url: "<?php echo base_url().'site/addFriend'?>",
        data: {
            idUser: friend
        },
        success: function () {
            $("#addFriend_" + friend).remove();
        }
    });
}
```

Figura 4.26 - Código *AJAX* para enviar um pedido de amizade

Para receber o parâmetro data no controlador é necessário fazer uma chamada que se encontra na Figura 4.27.

```
$this->input->post('idUser');
```

Figura 4.27 - Obter dados do *post*

4.3.3 *View* Mapa

Esta é a *view* responsável por apresentar os locais ao utilizador numa dada localização. Do lado esquerdo é apresentado um painel azul que é usado para filtrar os resultados em baixo são apresentados os locais com alguma informação importante para os utilizadores. Do lado direito é apresentado um mapa que contem os marcadores dos locais, ao passar o rato em cima do marcador é apresentado o local, bem como se passar o rato em cima do resultado do lado direito o mapa será aproximado a esse local para o utilizador visualizar a posição exata. Caso o mapa seja movido pelo utilizador serão apresentados novos resultados. Na Figura 4.28 é apresentado o mapa da aplicação.

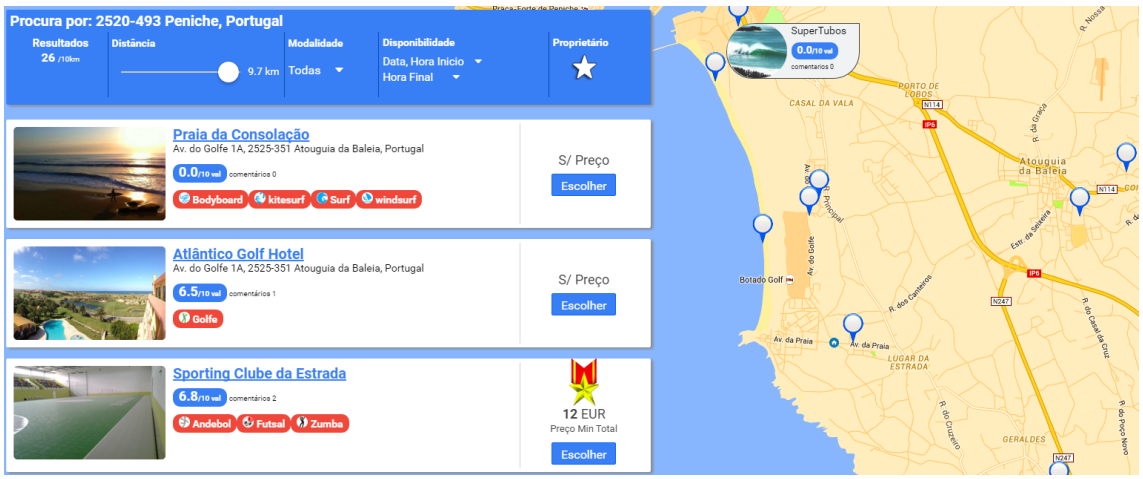

Figura 4.28 - Mapa da aplicação

Quando o utilizador aplica uma ação nesta página como mover o mapa para visualizar outra localização ou filtre por distância, modalidade, proprietário ou disponibilidade será apresentados ou removidos resultados. Para que a aplicação torne mais flexível foi necessário utilizar mais uma vez a tecnologia AJAX, pois seria fastidioso se cada vez que um utilizador movesse o mapa a aplicação teria de fazer um *refersh* para ser apresentados novos resultados. Ao contrario que acontecia na secção 4.3.2 a tecnologia AJAX era utilizada para fazer um *insert* na base de dados pois, era para enviar um convite de amizade a um determinado utilizador. Neste caso são será um *insert* mas sim um *select* para selecionar os locais e os seus dados com base no tipo de pesquisa que temos a fazer, no caso da Figura 4.28 é uma pesquisa por distância 9.7km sendo que os outros parâmetros como a modalidade não será filtrado.

```
function searchPlaces(lat, lng) {
    $.ajax({
        type: "POST",
        url: "<?php echo base_url().'site/redefineSearch'?>",
        data: {range: range, lat: lat, lng:lng,sport:sportValue,status:status,
            datetime_start:availability_start,time_end:availability_end, owner: star_active},
        dataType: 'json',
        success: function (res) {
            var i;
            $("#result_places").empty();
            for (i = 0; i < res.length; i++) {
                $("#result_places").append("<div class='row row_place_result place_result' id='" + i + "'>" +
                    "<div class='col col-25 place_image_result'>" +
                    "<img src='" + image + "' class='image_result' />" +
                    "</div>" +
                    "<div class='col col-55 display_block description_resultList'>" +
                    "<div><a href='"+dateON+"' class='positive text_place_result'><span class='name_resultlist'>"+
                    "<div><span id='local_" + i + "'><span id='local_current_"+i+"'></span></span></div>" +
                    "<div class='place_evolution_result'> <span class='badge badge-positive' style='font-size: 18px'>" +
                    "<span style='font-size: 10px; margin-left:3px;'>comentários " + res[i].nComments + "</span></div>" +
                    "<div class='place_evolution_result place_sports_result' id='sport_" + i + "'>" +
                    "</div>" +
                    "</div>" +
                    "<div class='col col-20 split-border-result text-center price_result display_block'>" +
                    owner_html+
                    "<div><span class='text-color-dark'style='font-size: 18px'>"+price+"</span><div>"+
                    "<div><span class='text-color-dark' style='font-size: 13px'>"+span_info_price+"</span><div>"+
                    "<div><a href='"+dateON+"' class='button margin-top-8 button_choose_place button-positive'>Escolher" +
                    "</a><div>"+
                    "</div>" +
                    "</div>"
                );
            }
        },
    });
}
```

Figura 4.29 - Código AJAX com jQuery para selecionar locais

Na Figura 4.29 mostra uma parte do código que é executado cada vez que um utilizador faz um filtro de dados ou movimenta o mapa. O *result_places* é a *div* responsável pelo armazenamento dos locais que irá fazer um *append* a cada local encontrado na base de dados numa distância de raio x.

4.3.4 *View* local

Esta é a *view* responsável por apresentar a página do local. Caso o local tenha prioritário poderá ser realizada uma reserva perante um evento. Na Figura 4.30 é apresentado o sistema de disponibilidade perante a data hora inicial e final

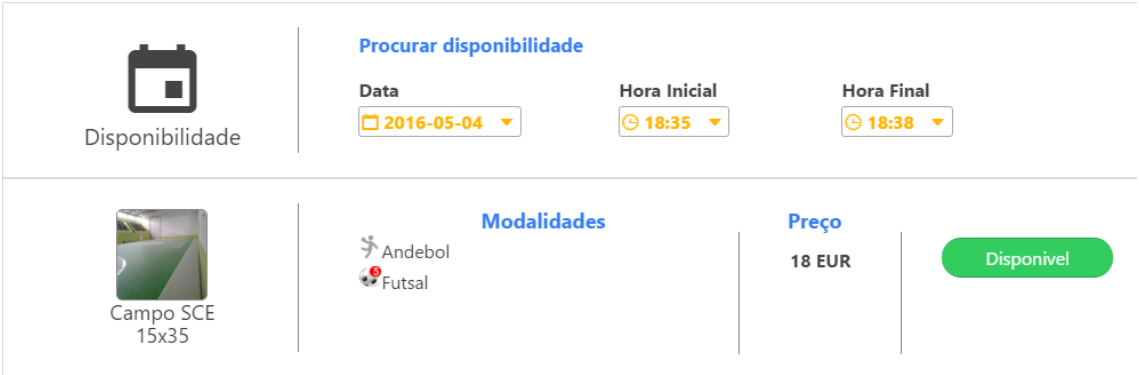

Figura 4.30 - Sistema de disponibilidade na página local

Este sistema também partilha a mesma tecnologia AJAX para evitar *refresh* à página. O proprietário do local pode adicionar, remover e editar dados dos espaços. Na Figura 4.31 está representado a edição de um espaço no local.

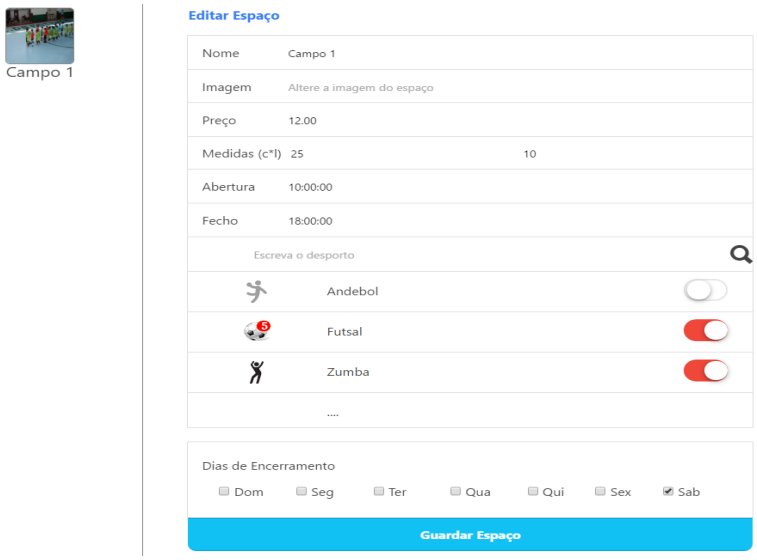

Figura 4.31 - Editar espaço do local

4.3.5 *View* registar local

Esta *view* é responsável por permitir que os utilizadores registem locais que ainda não exista na aplicação. Será apresentado um mapa ao utilizador e uma caixa de texto a pedir para ser introduzida a localização, depois será apresentada no mapa os *markers* de locais nessa localização que já estão registados na aplicação, desta forma o utilizador poderá verificar que o local que pretende adicionar não existe. O utilizador terá de fixar um *marker* que indica a localização do local e preencher os devidos campos como nome, contactos, fotografia e modalidades que podem ser realizadas. Se o utilizador for o proprietário do local poderá associar a sua conta a esse local, para gerir assim as suas reservar e os espaços. Na Figura 4.32 está apresentado a view em que o utilizador está a fixar o marcador na localização correta.

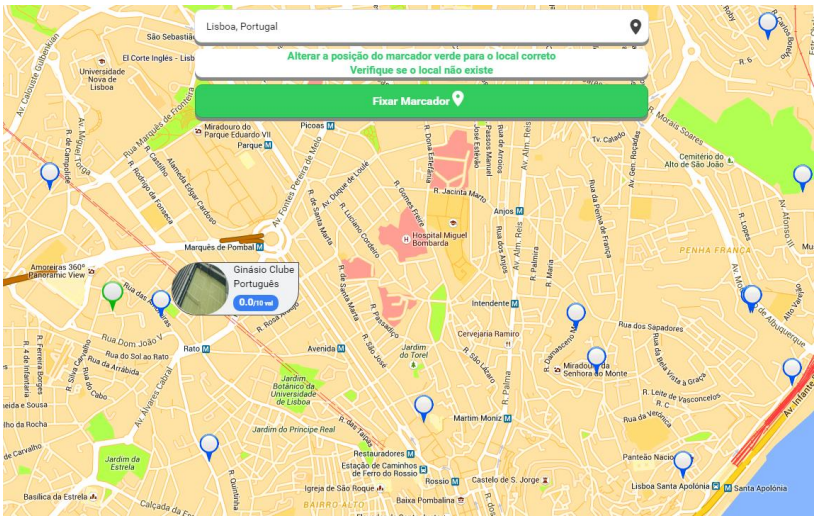

Figura 4.32 - Registar um local, marker verde. Locais adjacentes markers azuis.

4.3.6 *View* Mensagens (chat)

Esta *view* é responsável pela comunicação entre os membros da aplicação. É possível ao utilizador comunicar com qualquer membro da aplicação ficando a conversa sempre registada e dando a possibilidade de aceder a qualquer momento. Para aceder às conversas é disponibilizado uma lista de contatos com os utilizadores que já foram enviadas mensagens em conjunto com os amigos que o utilizador tenha, os contactos estão organizados de acordo com o ultimo utilizador que foi contactado. Na Figura 4.33 é apresentado a *view* do *chat*.

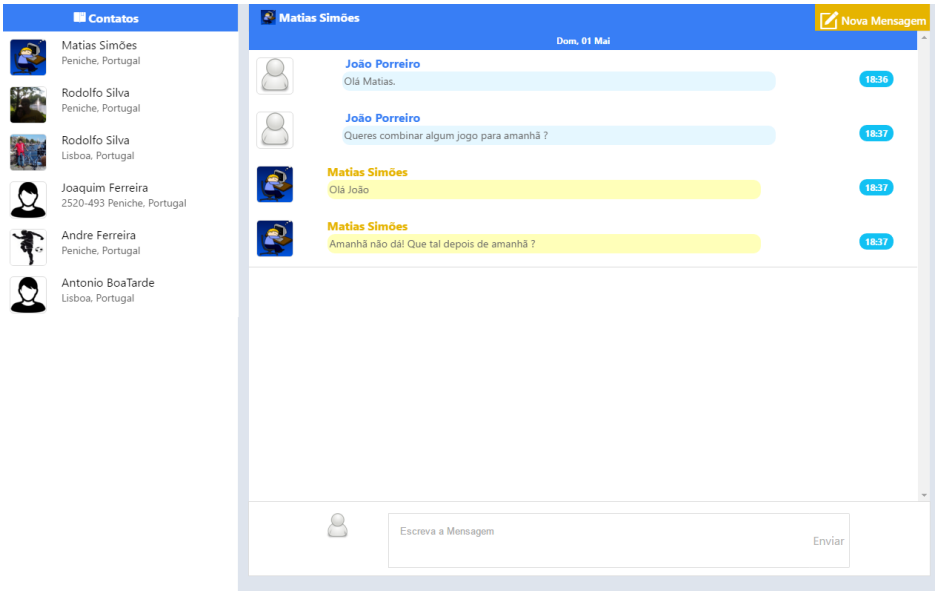

Figura 4.33 – View conversa de chat

A conversa no *chat* é praticamente em tempo real utilizado a tecnologia AJAX descrita na secção 4.3.3 , no entanto para que se torne uma conversa em tempo real é necessário a aplicação fazer vários *selects* à base de dados para mostrar novas mensagens aos utilizadores. Na Figura 4.34 mostrado o código responsável pela conversa dos utilizadores em tempo real.

```
setInterval(
    function(){
    searchMessages(); }, 4000
);
```

Figura 4.34 - Código JavaScript obter mensagens

O *setInterval* irá chamar a função *searchMessages()* que contem a tecnologia AJAX para selecionar dados da base de dados, de 4 em 4 segundos. Desta forma, dá a sensação que a conversa é praticamente em tempo real.

Quando o utilizador está em outra *view* e recebe uma mensagem o utilizador irá receber um aviso que pode ser visualizado na Figura 4.35.

Figura 4.35 - Avisos de mensagens recebidas

Estes avisos de mensagens recebidas são as fotografias dos utilizadores com o número de mensagens novas que enviou. A tecnologia utilizada é idêntica a do sistema de *chat.*

4.3.7 *View* inicial equipa

Esta *view* é como uma sala de convívio virtual para os utilizadores que pertencem a uma equipa, que no qual dá a possibilidade de fazer *posts* e comentários aos *posts* criados. Os utilizadores podem dar feedback dos *posts* deixando uma star (equivalente ao *like* do Facebook). É de notar com este sistema de *posts* os utilizadores da equipa podem combinar as horas e os locais dos duelos a fim de chegarem a um consenso. Quando existe um novo *post* ou comentário todos os elementos da equipa são notificados de forma, a que estejam atentos às novidades. Na Figura 4.36 está apresentado o sistema de *posts* numa determinada equipa.

Figura 4.36 - Sistema de posts da view página da equipa

4.3.8 *View* eventos

Esta *view* é responsável pela gestão de um determinado evento. Nesta *view* um utilizador poderá visualizar a toda informação relativa a horários e local e visualizar quais os membros que vão estar presentes, caso estiver convidado poderá juntar à equipa ou rejeitar o convite. O criador do evento pode convidar os membros da equipa que pretender e cancelar o evento se ninguém estiver interessado no evento. Na Figura 4.37 está representado a *view* de um evento tipo duelo onde, o utilizador é o criador do evento e o duelo ainda não foi aceite pela equipa adversária podendo ser alterada até não obtiver resposta, caso não aceite poderá também ser escolhida outra equipa adversária.

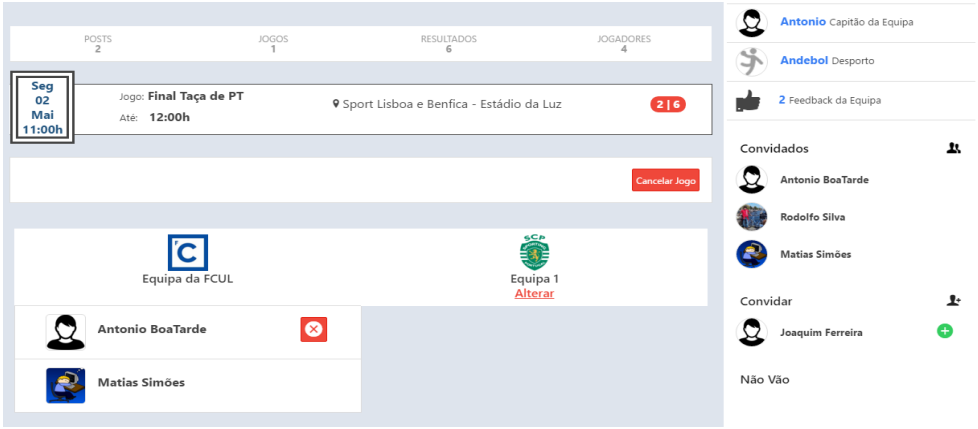

Figura 4.37 - *View* do evento tipo duelo

Quando um elemento é convidado é inserido na base de dados em estado de convite, o elemento convidado ao aceder à página do evento irá deparar-se com o que está na Figura 4.38.

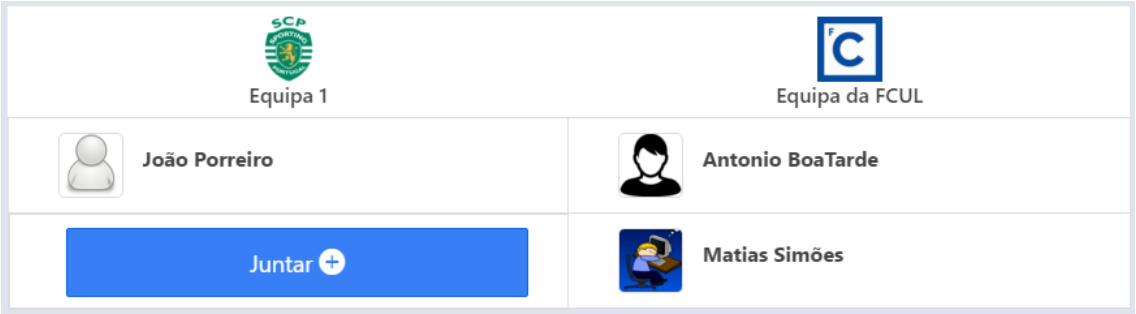

Figura 4.38 - *View* do evento no estado "convidado" ou "não vou"

O utilizado quando clica no "Juntar" é executado um *update* à base de dados deixando o utilizador no estado "vou". A tecnologia utilizada é novamente em AJAX de forma a evitar o *refresh* na página. O utilizador poderá cancelar o convite mesmo que esteja no estado "vou" alterando o estado para "não vou" e podendo ser novamente alterado de "não vou" para "vou", no entanto, os jogos tem o número de elementos máximo, caso seja atingido o utilizador não poderá comparecer. A Figura 4.39 apresenta o estado do utilizador depois de clicar no botão "Juntar".

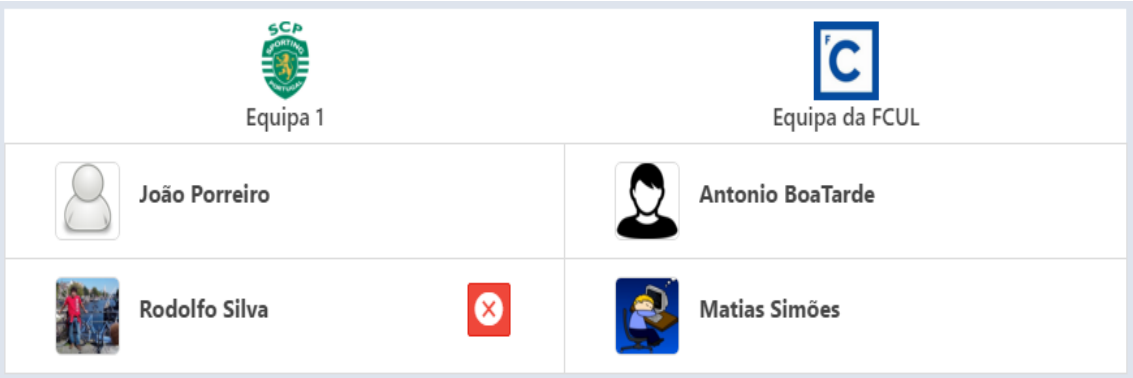

Figura 4.39 - *View* evento no estado "vou"

4.3.9 *View* resultados equipa

Esta *view* é responsável por apresentar os resultados dos jogos da equipa e em caso de eventos tipo duelo os capitães podem dar *feedback* às equipas adversárias e apresentar o inserir o resultado do jogo. A primeira equipa a inserir o resultado do jogo será dada a sugestão de resultado neste caso, é executado um *insert* na base de dados com estado de pendente a equipa adversária quando verificar o resultado poderá concordar que aquele foi o resultado neste caso, é executado um *update* à base de dados alterando o estado para aceite ou recusar passando a base de dados para o estado recusado. Na Figura 4.41 é apresentado a inserção do resultado e o *feedback* à equipa adversária, na Figura 4.40 é apresentado a equipa adversária a responder ao resultado deixado pela primeira equipa.

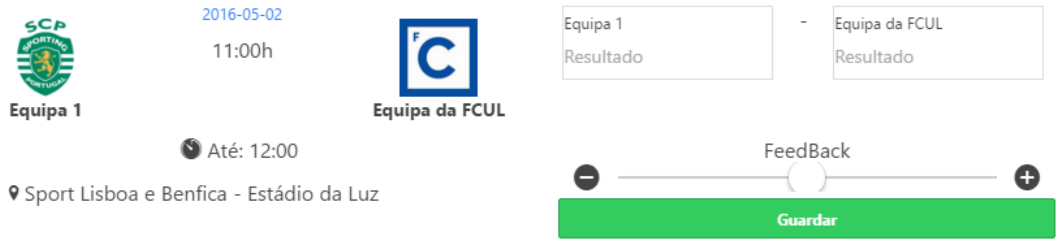

Figura 4.41 - *View* resultado, inserir o resultado do duelo

Figura 4.40 - *View* resultado, aceitar o resultado do duelo

4.3.10 *View* convidar jogadores para equipa

Esta *view* é responsável pelo convite de novos membros para a equipa. Um utilizador com privilégios na equipa pode pesquisar por um membro na aplicação que ainda não esteja em estado de convite ou já afeto à equipa. Na Figura 4.42 é apresentada a respetiva *view* com um utilizador a convidar um membro.

Figura 4.42 - *View* convite de jogadores numa equipa

Na Figura 4.42 poderá visualizada no painel do lado direito os convites que foram enviados na secção "convidados" e os convites que foram recebidos, na secção "Aceitar Jogadores" para ingressar na equipa. Para evitar os *refresh* na página é utilizada a tecnologia AJAX nestas operações. No campo de pesquisa de procura de utilizadores à medida que o utilizador insere um caracter é filtrado as previsões, sendo que estão ordenadas pela localização. Para realizar esta funcionalidade é apresentado o código responsável por efetuar a procura.

```
public function getAutoTeamComplete($lat, $lng, $idTeam, $name)
{
    $query = $this->db->query('SELECT image,name,id,location,user_name,
                                    (3959 * acos (cos(radians(?)) *
                                    cos(radians(lat)) *
                                    cos(radians(lng) - radians(?)) +
                                    sin (radians(?)) * sin(radians(lat))))
                               AS distance FROM user
                               LEFT JOIN userteamlist ON idUser = user.id and idTeam = ?
                               WHERE idUser IS NULL and user.location != "" AND user.name LIKE ?
                               ORDER BY distance', array($lat, $lng, $lat, $idTeam, $name . '%'));
    return $query;
}
```

Figura 4.43 - Código de procura de elementos para equipa

A função em *mysql* referida em cima recebe como parâmetros a latitude, longitude, o id da equipa e o nome que está a procurar. No *select* é efetuado o cálculo da distância

com base na latitude e longitude para poder organizar os utilizadores por distância. O *left join* apresentado é responsável por retirar dessa pesquisa os membros que já estão dentro da equipa e os que já estão em estado de convidado. É de notar que cada vez que é inserido um determinado caracter na caixa de texto de pesquisa de jogadores é executada esta função *mysql* para filtrar as previsões, de forma a facilitar a pesquisa.

Capítulo 5

Testes

Neste capítulo serão apresentados e descritos os resultados dos testes de usabilidade da aplicação.

5.1 Testes de Usabilidade

Partindo da Engenharia de Software a usabilidade é englobada dentro da qualidade e visa avaliar a facilidade dos utilizadores a utilizarem as interfaces. De acordo com a norma ISO 9241 [8] a aplicação web deve seguir as seguintes componentes de usabilidade:

- **Aprendizagem** – Quão fácil é para o utilizador realizar tarefas na primeira vez que usa a aplicação.
- **Eficiência** – Depois dos utilizadores "conhecerem" a interface da aplicação, o quão rápido eles conseguem voltar a realizar tarefas.
- **Memória** – Depois de um utilizador voltar a usar a aplicação, e após algum tempo de inatividade, o quão é fácil conseguir realizar as tarefas.
- **Satisfação** – Quão agradável é para o utilizador usar a aplicação.

Para além das componentes apresentadas na norma ISO 9241 a usabilidade também pode ser especificada sobre outras perspetivas. Para isso também deverá ser seguido as dez heurísticas de Nielsen [9] para o *design* da interface:

1. **Visibilidade do estado da aplicação –** A aplicação deve manter sempre o utilizador informado do que está a ocorrer, através de *feedback* apropriado num prazo razoável.

2. **Relação entre a aplicação e mundo real –** A aplicação deve "falar" a linguagem do utilizador com palavras, frases e conceitos que sejam familiares para o

utilizador. Deve-se seguir convenções do mundo real, fazendo com que as informações apareçam numa ordem lógica e natural.

3. **Controlo do utilizador e liberdade –** Os utilizadores costumam escolher as funções do sistema por engano e desta forma é necessária uma "saída de emergência" de forma a deixar o estado indesejado sem ter que passar por um diálogo longo. O sistema deve suportar *undo* e *redo*.

4. **Consistência e padrões -** Os utilizadores não devem ter a preocupação se palavras, situações ou ações diferentes significam o mesmo. Deve-se seguir as convenções da plataforma.
5. **Prevenção de error –** Melhor que as mensagens de erro é um *design* cuidadoso que impede a ocorrência de um problema. Deve-se eliminar as condições possíveis de erros ou verifica-los, apresentado aos utilizadores uma opção de confirmação antes de se comprometerem com uma determinada ação.

6. **Reconhecimento em vez de recordação –** Minimizar a memória do utilizador tornando objetos, ações e opções visíveis. As instruções para a utilização da aplicação devem estar visíveis ou facilmente recuperáveis sempre que necessário.

7. **Flexibilidade e eficiência de utilização –** A aplicação deve ser fácil e eficiente de usar pelos utilizadores, sejam estes inexperientes e/ou experientes. Deve-se fornecer "teclas de atalho" ou "funções" para que, com a crescente experiência de utilização, os utilizadores experientes consigam navegar de uma forma mais eficiente na aplicação de modo a realizarem as tarefas mais frequentes.

8. **Estética e *design* minimalista –** Os diálogos não devem conter informações que são irrelevantes ou desnecessárias. Cada unidade extra de informação num diálogo compete com as unidades de informação relevantes diminuindo assim a sua visibilidade relativa.

9. **Ajudar os utilizadores a reconhecer, diagnosticar e recuperar de erros –** Ajudar os utilizadores a reconhecer diagnosticar e recuperar de erros que possam acontecer na aplicação.

10. **Ajuda e documentação –** Mesmo que a aplicação seja capaz de ser utilizado sem ajuda de documentação, poderá ser necessário fornecer ajuda e documentação ao utilizador caso este tenha dúvidas em realizar uma alguma tarefa.

5.2 Testes de Usabilidade Presencial

Nos testes de usabilidade o utilizador foi convidado a interagir com aplicação seguindo um conjunto de tarefas. Em cada tarefa realizada o moderador recolhia as informações dadas pelo utilizador. Na realização dos testes foi elaborado um questionário, em que este está dividido:

- **Detalhes das modalidades praticadas:** nesta secção é solicitado ao utilizador que indique se pratica alguma modalidade desportiva e se utiliza alguma rede social, dado que as interfaces da aplicação são idênticas às redes sociais, logo é calculado que os utilizadores que tenham terão mais facilidade a interagir.
- **Tarefas:** na Tabela 5.1 é dado um conjunto de tarefas em que para cada tarefa o utilizador teria de classificar a tarefas por facilidade numa escala de um (muito fácil) a cinco (muito difícil). Caso o utilizador não conseguisse concluir a tarefa com sucesso era permitido atribuir a tarefa como "não concluída".

Avaliação das Tarefas

Uma tarefa é registada como sucesso, caso o utilizador a consiga concluir. Caso o utilizador não consiga realizar uma tarefa é assinalada como "Não conclui" caso contrario o utilizador é convidado a escolher o grau de facilidade na concretização atribuindo uma escala numérica de um a cinco: um (muito fácil), dois (fácil), três (razoável), quatro (difícil), cinco (muito difícil). Em ambos os casos de sucesso ou insucesso na concretização de uma tarefa o utilizador pode dar uma sugestão referindo formas de melhorar. A Tabela 5.1 apresenta as tarefas que foram realizadas pelos utilizadores.

Tarefa	Descrição
1	Na página inicial faça um registo de um novo utilizador.
2	Na próxima página preencha os dados que são pedidos e inclua na escolha das modalidades o andebol e outras que goste.
3	Crie uma equipa preenchendo a informação que lhe é pedida e escolha como modalidade o andebol.
4	Para convidar elementos para sua equipa procure pelo seu amigo “João Porreiro”.
5	Faça uma pesquisa na aplicação e tente encontrar a equipa “Atlético Sport” e entre na página.
6	Envie um convite para a ingressar na equipa.
7	Volte à página inicial e entre nas mensagens e envie uma mensagem ao “João Porreiro” a referir que o adicionaste na tua equipa.
8	Vá ao menu e faça *logout* na aplicação.
9	Faça um login na conta do João. Em que o utilizador é “joao” e a *password* é “12345”.
10	Visualize as notificações recebidas do João.
11	Visualize os convites. Aceite o convite para ingressar na equipa que criou e aceite o convite que enviou para a equipa “Atlético Sport”.
12	Responda à mensagem que escreveu anteriormente. “Já aceitei”.
13	Entre na equipa “Atlético Sport” visualize os comentários do primeiro post e faça um comentário "Não vou poder ir".
14	Visualize os resultados e os jogadores da equipa.
15	Ainda na página da equipa crie um evento tipo treino preenchendo a informação do treino com o nome, data, hora e com um máximo de doze jogadores. Convide quatro jogadores para o treino, incluindo o "João Porreiro".
16	Procure o local “Sporting Clube da Estrada” para praticar a modalidade e entre na página do local.
17	Na página do local visualize a informação as fotografias e as avaliações.

18	Verifique na secção da disponibilidade os campos disponíveis para essas datas. Caso exista disponibilidade reserve um dos campos, caso contrário pesquise outra data e faça a reserva.
19	Entre dentro da página do evento e escolha jogar pela Equipa B.

Tabela 5.1 - Tarefas do questionário de usabilidade

Na Tabela 5.2 estão representados os resultados esperados dos testes realizados pelos utilizadores.

Tarefa	Esperado
1	Preenchimento dos campos "nome de utilizador", "*email"* e "*password"* e clicar no botão "registar"
2	Preenchimento dos campos primeiro e ultimo nome, "localização" e escolher na *checkbox* as modalidades incluindo andebol
3	Na pagina inicial clicar no botão "Criar equipa" e preencher os campos nome, localização e na *checkbox* escolher a modalidade.
4	Na página da equipa criada clicar no botão "Convidar jogadores" e no campo "Procurar jogador" digitar o nome "João Porreiro" e clicar no botão de adicionar.
5	No cabeçalho da aplicação pesquisar no campo "Pesquisar localizações, jogadores, equipas, locais" a equipa pelo termo "Atlético Sport"
6	Na página da equipa "Atlético Sport" na secção "Aderir à equipa" clicar no botão "Enviar Convite".
7	Clicar no botão "inicio" e na página de inicio clicar no botão "Mensagens" pesquisar pelo nome "João Porreiro" no campo "procurar utilizadores" e escrever no campo "escrever mensagem" que já adicionou o João à nova equipa
8	Clicar no botão menu no canto superior direito e clicar no botão "Logout"
9	No formulário de login digitar "Joao" no campo "Utilizador" e "12345" no campo "*password*".
10	No cabeçalho da aplicação clicar no símbolo da campainha para visualizar as notificações

11	No cabeçalho da aplicação clicar no símbolo da estrela para aceitar o convite para a equipa que criou clicando no botão adicionar e aceitar o convite para a equipa "Atlético Sport" que enviou no botão adicionar.
12	No canto inferior direito clicar na imagem do utilizador e na página de mensagens inserir a mensagem no campo "Escrever mensagem" "já aceitei".
13	Voltar à página de inicio escolher a equipa "Atlético Sport" e na secção dos "Posts" comentar a primeira publicação escrevendo no campo "Responder" "não posso ir".
14	Para visualizar abrir a secção "Resultados" e abrir a secção "Jogadores".
15	Clicar no botão "Criar Evento" e de seguida clicar na secção "Treino" na secção do treino preencher os campos "nome do treino", "N. max de jogadores" e escolher na *checkbox* quatro membros incluindo o "João Porreiro" e clicar "Atribuir Localização".
16	Procurar na lista de locais o local "Sporting Clube da Estrada".
17	Na pagina da equipa visualizar informação, passar o rato nas imagens miniaturas para visualizar as imagens em grande e fazer *scroll* até ao final da página para visualizar os cometários.
18	Na secção "Procurar Disponibilidade" escolher o espaço clicando no botão "Reservar"
19	Na lista de jogos clicar no nome do jogo para abrir a página do jogo. Na secção da "Equipa 1" remover o utilizador clicando no botão remover e clicar no botão "Juntar" na secção da "Equipa 2".

Tabela 5.2 - Tabela do que é esperado por cada tarefa

Resultados

Foram inquiridos doze utilizadores com idades a partir dos vinte e três anos até aos cinquenta e cinco, que se disponibilizarem a realizar os testes. Todos os utilizadores gostam de fazer desporto sendo, que 67% pratica uma modalidade coletiva. Apenas um utilizador não utilizava as redes sociais.

Na Figura 5.1 é apresentado um gráfico que indica as tarefas concluídas pelos utilizadores em percentagem. Foi possível concluir um dos utilizadores não conseguiu encontrar os botões para visualizar as notificações e os convites que representam as tarefas 10 e 11 respetivamente, dado que este utilizador não utilizava redes sociais que no

qual essas funcionalidades são bastante ativas para quem usa redes sociais. Um outro utilizador não conseguiu reservar o local (tarefa 18) e apresentou como argumento que não conseguiu encontrar disponibilidade, dado que era um dos últimos utilizadores a realizar o teste os primeiros utilizadores já teriam reservado os espaços/campos do local nos próximos dias e horas, como resultado o utilizador não conseguia encontrar disponibilidade pelo que teria de pesquisar por datas mais futuras. Como o utilizador não conseguiu concluir a tarefa 18 era impossível concluir a tarefa 19 pois, precisava da reserva do local para conseguir entrar na página do jogo.

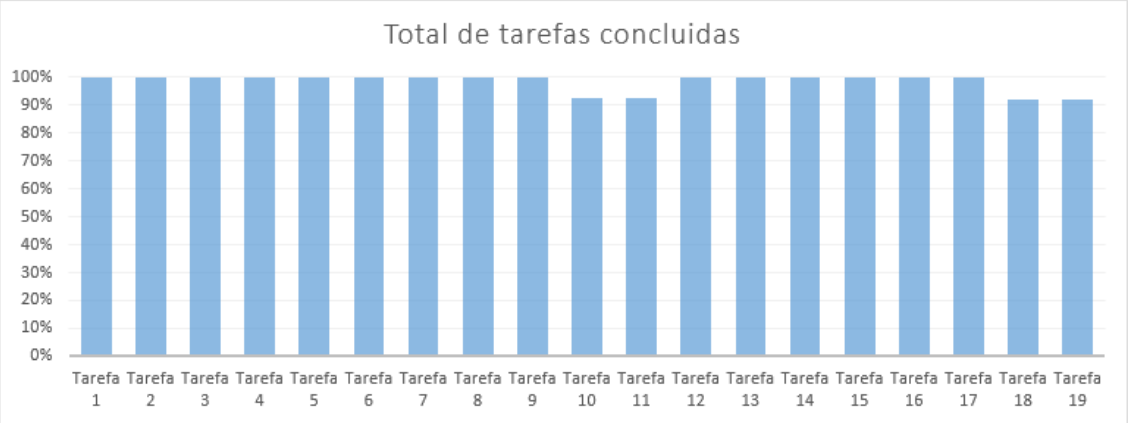

Figura 5.1 - Tarefas concluídas em percentagem

Na Figura 5.2 é representado o gráfico expressa a media de dificuldades dos utilizadores perante cada tarefa. Estes resultados adquiridos têm apenas em conta os utilizadores que conseguiram completar uma dada tarefa ou seja, caso o utilizador não consiga completar a tarefa não poderia expressar a sua dificuldade.

Figura 5.2 - Escala de dificuldade de cada tarefa

Embora os utilizadores não tenham expressado grandes dificuldades em realizar os testes foram deixadas algumas notas importantes como:

- **Tarefa 2 –** Em caso de erro no formulário os utilizadores teriam de inserir novamente os resultados. Foi utilizado uma função php *setvalue()* para recuperar os dados em caso de erro.
- **Tarefa 3 –** Na criação da equipa existe uma *checkbox* para selecionar a modalidade da equipa. Os utilizadores fizeram a sugestão de alterar a seleção de vermelho para verde pois o vermelho dava a sensação contrario à seleção.
- **Tarefa 10 e 11–** Dois dos utilizadores indicaram que seria mais fácil de encontrar as notificações ou os convites caso indicasse o número de notificações ou convites novos.

Figura 5.3 - Sugestão tarefa 10 e 11

- **Tarefa 18 –** Ao pesquisar pela disponibilidade de um local caso não existisse disponibilidade não apresentava feedback. Os utilizadores sugeriram para apresentar uma mensagem referindo que não existia disponibilidade.

Capítulo 6

Conclusão

A pratica de desporto sempre teve diversos benefícios importantes para a saúde e bem-estar. A rede social Sport4Stars vem facilitar atletas e futuros desportistas a encontrar locais e parceiros interessados em praticar o desporto e descobrir facilmente equipas adversárias para defrontar.

Com base no estudo realizado sobre aplicações semelhantes foi concluído que nenhuma das aplicações conseguia satisfazer na totalidade as necessidades dos utilizadores. Não existe nenhuma aplicação que torne possível a criação de equipas, que possua uma página onde os utilizadores possam publicar e comentar mensagem, que permita receber diversas sugestões de utilizadores para se juntarem à equipa, onde possibilite formarem torneios e duelos com diversas equipas e inserir os seus resultados na página. Outras das grandes funcionalidades presentes e inovadoras e que ainda não tem existência em outras aplicações é o facto de facilitar a reserva de um espaço num determinado local garantindo a disponibilidade da equipa para a realização de um evento desportivo. Algumas das aplicações estudadas tinham alguns problemas de desempenho que no qual prejudicava a sua utilização como a procura de eventos demorava muito tempo a dar uma resposta, além dos problemas de desempenho algumas das aplicações não funcionavam corretamente em termos de design responsivo o que acaba por prejudicar a usabilidade em dispositivos móveis como em *tablets* e *smartphones*.

Para o desenvolvimento da aplicação foi utilizado a *freamwork* de *php CodeIgniter*. O *CodeIgniter* utiliza o padrão MVC que contribui para uma maior estruturação e organização do código, além disso contem um conjunto de funções pré-definidas que ajudam em certas tarefas, como é o caso da verificação da inserção de formulários incorretos. Para auxiliar o design gráfico da aplicação também foi utilizado uma *freamwork* chamada *ionic.* O *ionic* contem um conjunto de estilos que permitem realizar um bom design gráfico tornando mais apelativo com a utilização dos botões, formulários, caixas de texto já pré-definidas. Com o crescente aumento dos dispositivos móveis tornou-se fundamental que as aplicações se adaptem a todos os dispositivos. Desta

forma, era fundamental a aplicação funcionasse em todos os dispositivos utilizando o método *Responsive Web Design*. A utilização do *ionic* facilitou a tarefa de criar a aplicação como método *Responsive* pois, possibilita a construção de uma *grid* que permite que uma determinada interface se adapte mais facilmente ao comprimento do dispositivo. A escolha do design da aplicação foi muito estudada, além de ser necessário uma interface com um bom aspeto era fundamental que os utilizadores conseguissem facilmente utilizar a aplicação desde a primeira interação.

Neste projeto houve uma fase de recolha de informações de 804 infraestruturas desportivas, representado quase na totalidade o número de infraestruturas desportivas disponíveis em todo o país, sendo possível realizar diversas modalidades desportivas. Um utilizador de um ponto do país pode facilmente encontrar o local mais próximo para realizar a sua modalidade desportiva e informar-se relativamente às condições que apresenta. Caso a infraestrutura não esteja guardada na aplicação o utilizador se tiver conhecimento pode registar, preenchendo a informação que tem conhecimento.

Á medida que a aplicação foi construída foram surgindo novas ideias, como é o caso do local em o proprietário que pode gerir reservas, adicionar, remover espaços ao local permitindo que os utilizadores efetuassem pesquisas de locais com base na disponibilidade. Também foi decidido que a aplicação deveria ter a pagina da equipa para guardar os resultados dos eventos e publicar mensagens para os restantes utilizadores da equipa e por fim, foi definido que cada utilizador deveria poder ter a sua página que apresentasse as equipas do utilizador, jogos a realizar, os resultados de todos os jogos que esteve presente e permitisse adicionar amigos para poder mais facilmente conversar no chat e visualizar as páginas de perfil. O registo de locais já tinha sido idealizado desde o inicio, no entanto era importante que esta funcionalidade surgisse para a aplicação continuar a evoluir permitindo que os utilizadores adicionem locais à aplicação de forma, a que os utilizadores possam planear eventos sobre esse local.

Os testes de usabilidade foram fundamentais e foram concluídos com sucesso, dado que todos os utilizadores conseguiram praticamente concluir todas as tarefas que lhes foram apresentadas, tendo sido avaliada no geral com o grau de dificuldade de muito fácil de utilizar. Estes testes permitiram ainda corrigir alguns problemas que foram encontrados pelos utilizadores que se disponibilizam a realizar o teste, dado que estes apresentaram algumas sugestões importantes para melhorar as interfaces da aplicação. Concluindo esta aplicação que foi desenvolvida no contexto na tese de mestrado de Engenharia Informática é uma grande utilidade para os praticantes de modalidades desportivas e principalmente para as pessoas que não praticam modalidades porque, não conhecem os locais ou não consigam encontrar elementos para formar equipas.

Apêndice A

Sistema de Diagramas de Sequência (SSD)

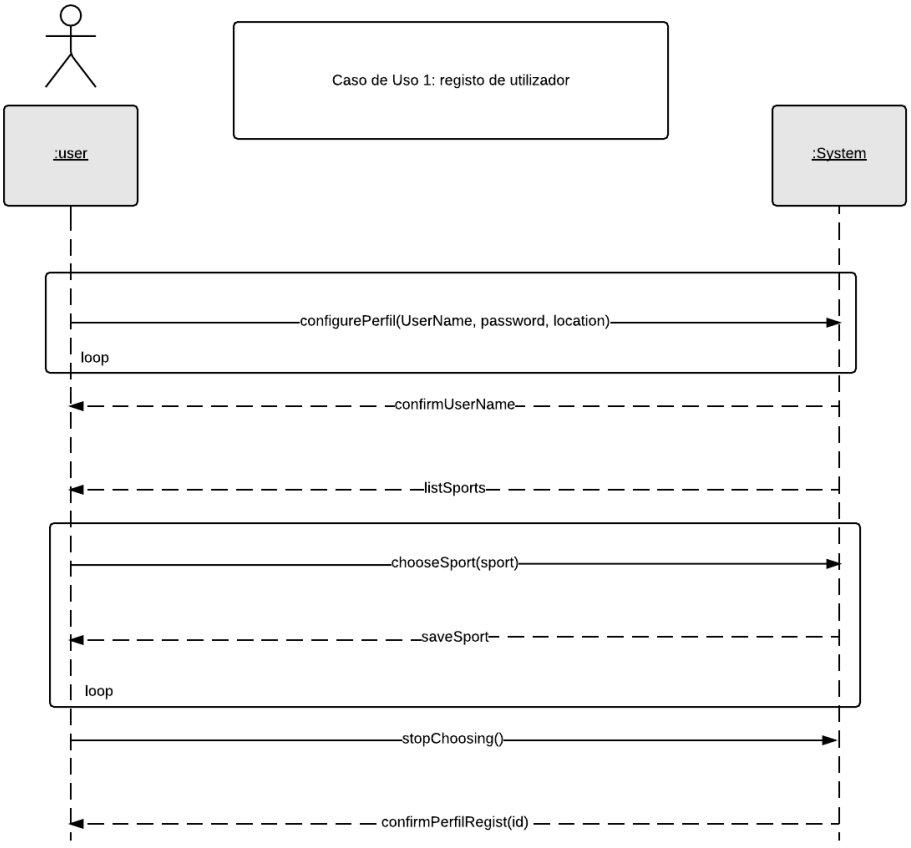

Figura A.1 - SSD registar utilizador

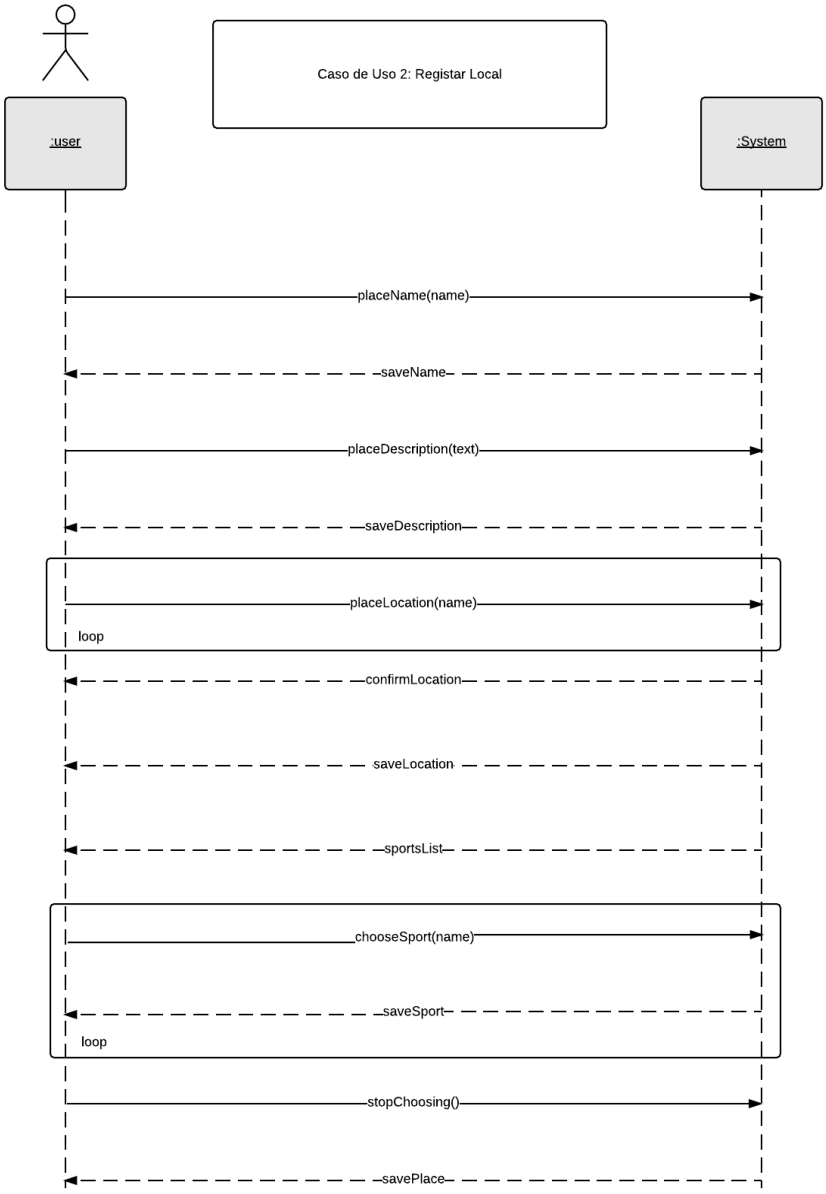

Figura A.2 - SSD registar local

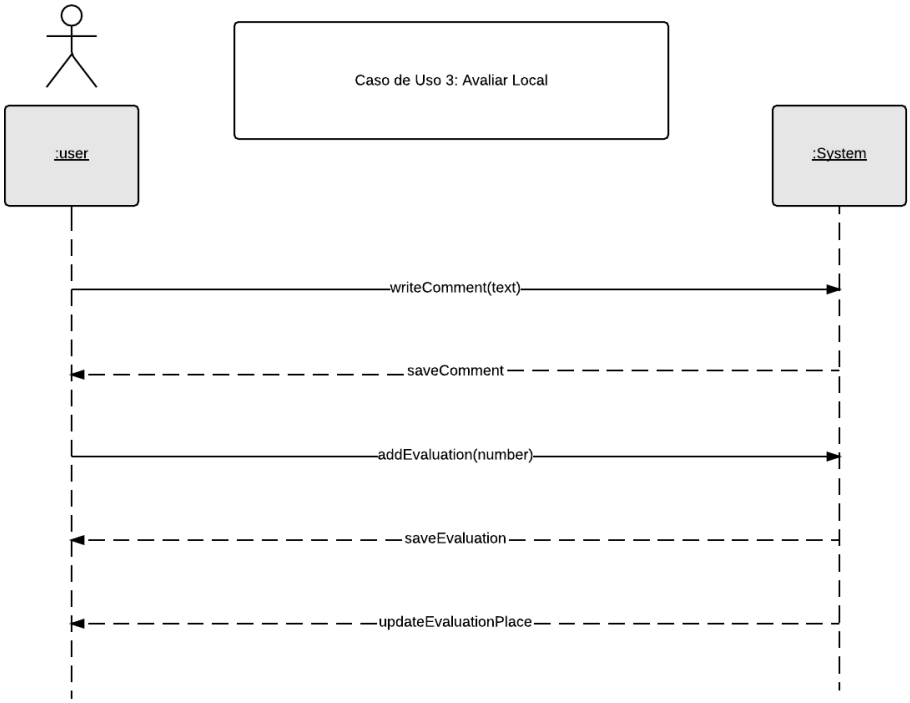

Figura A.3 - SSD avaliar local

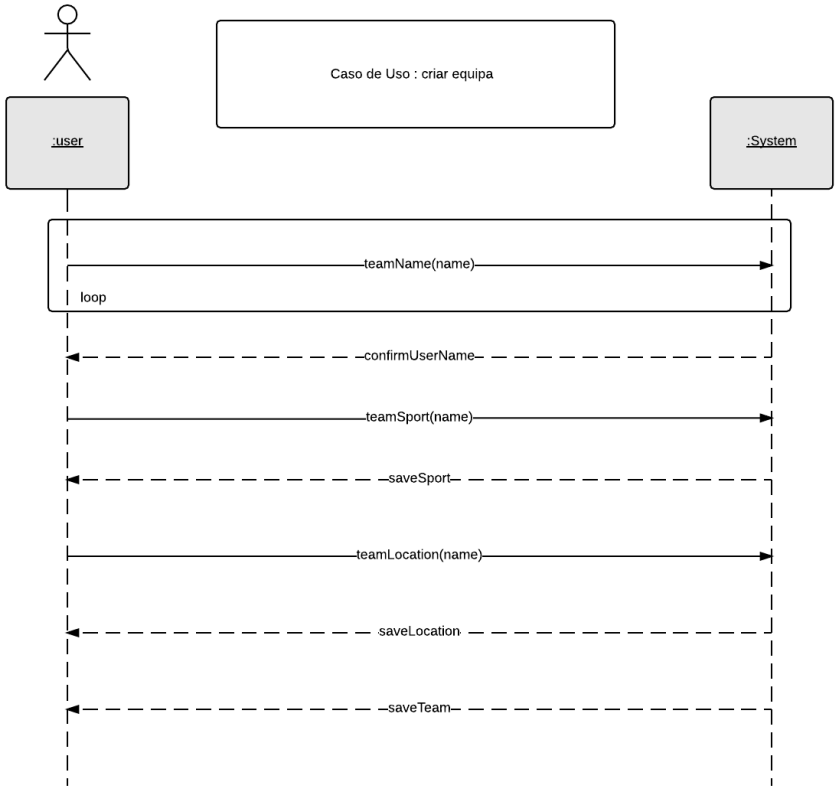

Figura A.4 SSD criar evento

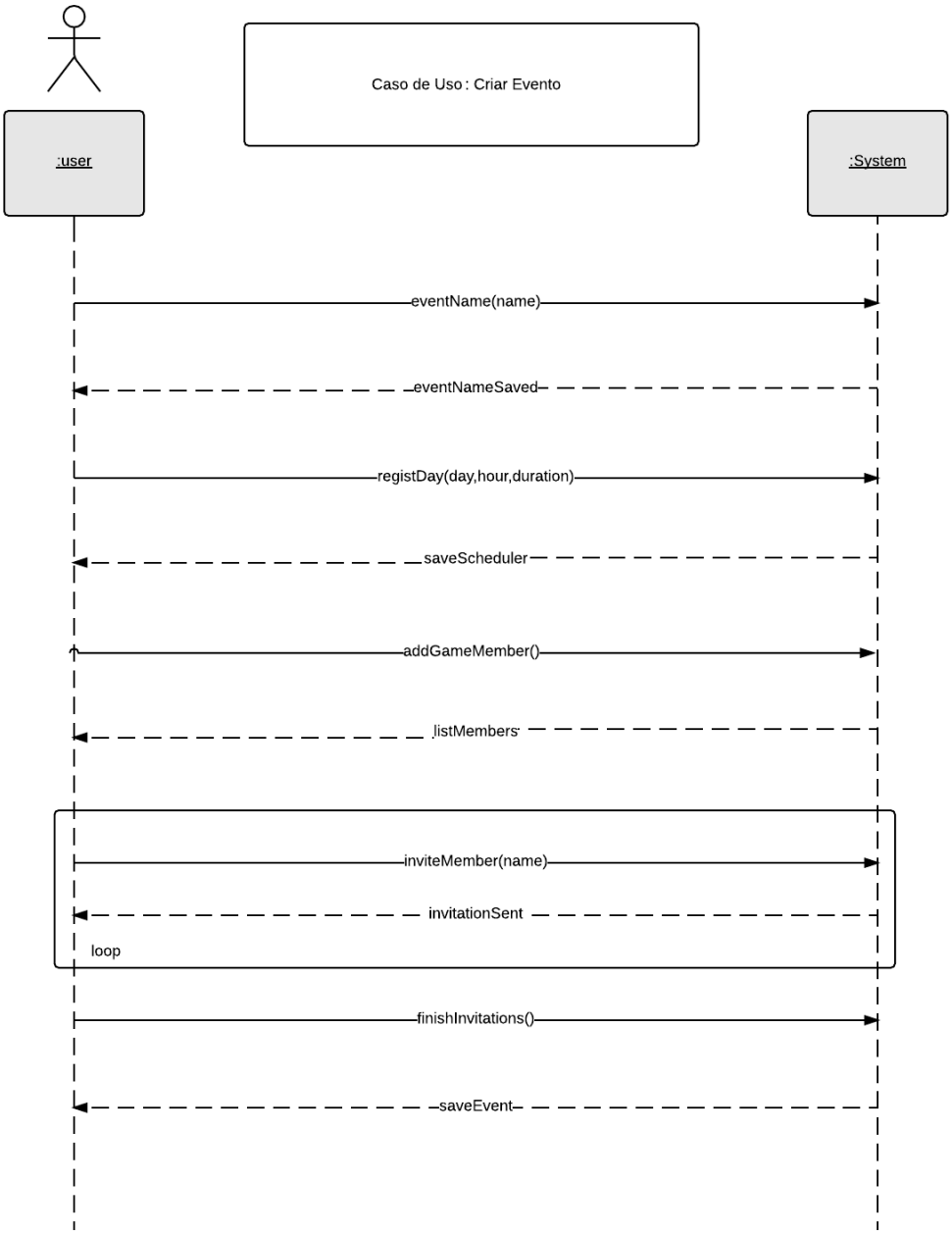

Figura A.5 - SSD criar evento

Apêndice B

Base de Dados

Para a construção da aplicação foi necessário construir uma base de dados relacional que irá permitir armazenar todos os dados relativos à aplicação web. Na Figura B.1 é representado o modelo de dados da aplicação.

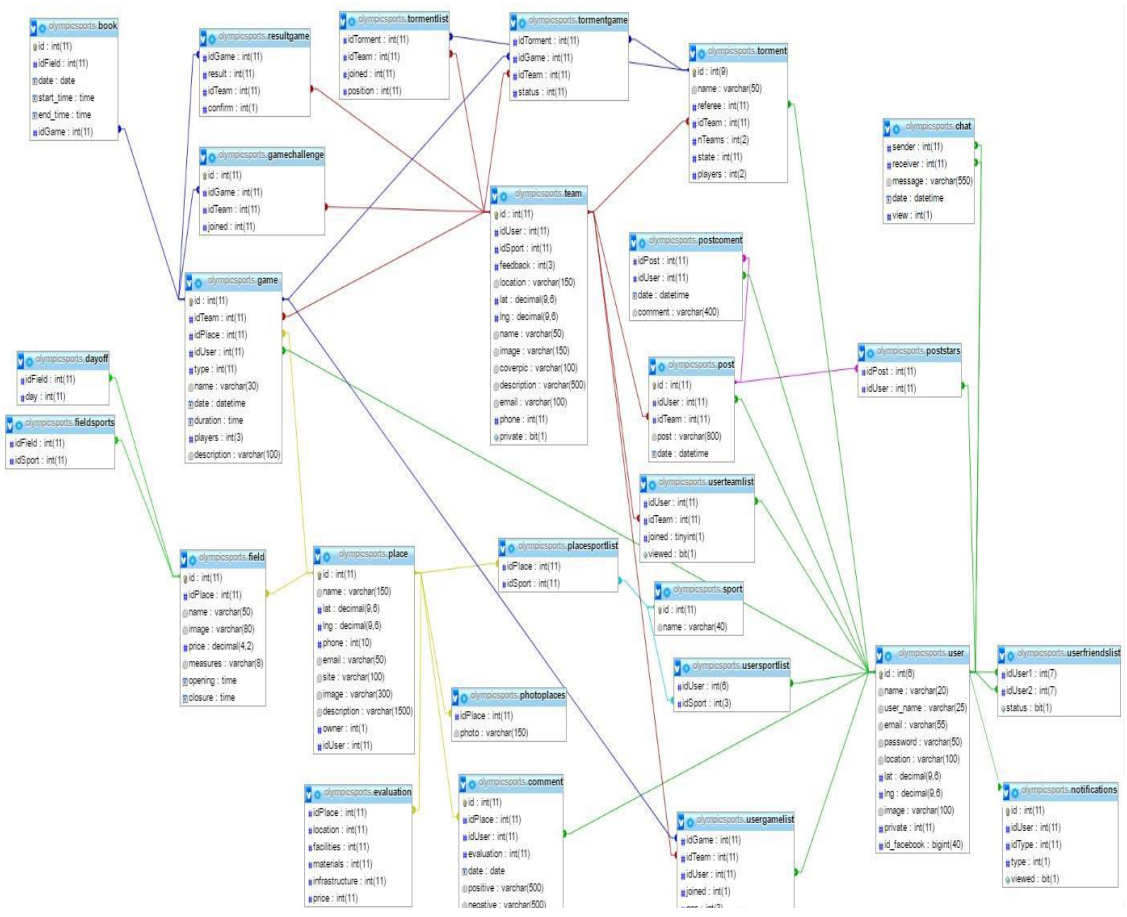

Figura B.1 – Modelo de dados

A base de dados é composta por 27 tabelas, nomeadamente:

- Tabela “user” é a tabela responsável por armazenar a informação dos utilizadores.
- Tabela “sport” é a tabela responsável por armazenar os dados dos desportos da aplicação.
- Tabela “usersportlist” é a tabela responsável por referenciar os desportos que um dado utilizador segue.
- Tabela “team” é a tabela responsável por armazenar os dados das equipas.
- Tabela “userteamlist” é a tabela responsável por referenciar os utilizadores que estão afetos a uma dada equipa.
- Tabela “post” é a tabela responsável por armazenar as publicações dos utilizadores nas equipas.
- Tabela “poststars” é a tabela responsável por referenciar os utilizadores que já adicionaram uma star numa publicação.
- Tabela “postcoment” é a tabela responsável por armazenar os comentários das publicações.
- Tabela “place” é a tabela responsável por armazenar os dados dos locais da aplicação.
- Tabela “placesportlist” é a tabela responsável por referenciar os dados dos desportos com o do local.
- Tabela “photoplaces” é a tabela responsável por armazenar as fotografias da galeria dos lugares.
- Tabela “comment” é a tabela responsável por armazenar os comentários referente aos locais escritos pelos utilizadores.
- Tabela “evaluation” é a tabela responsável por armazenar a avaliação dos locais avaliados pelos utilizadores.
- Tabela “field” é a tabela responsável por armazenar os espaços/campos que um lugar pode ter.
- Tabela “fieldsports” é a tabela responsável por referenciar os vários desportos que cada espaço pode conter.
- Tabela “dayoff” é a tabela responsável por referenciar os vários dias da semana que o espaço está fechado.
- Tabela “game” é a tabela responsável por armazenar os jogos das equipas da aplicação.
- Tabela “gameuserlist” é a tabela responsável por referenciar os utilizadores das equipas que vão, não vão ou estão convidados para um dado jogo.
- Tabela “gamechallenge” é a tabela responsável por armazenar as equipas quando existe um duelo ou um jogo de torneio.

- Tabela “resultgame” é a tabela responsável por armazenar os resultados das equipas dado um jogo.
- Tabela “torment” é a tabela responsável por armazenar os dados dos torneios da aplicação.
- Tabela “tormentlist” é a tabela responsável por referenciar as equipas participantes do torneio.
- Tabela “tormentgame” é a tabela responsável por referenciar os jogos existentes num torneio.
- Tabela “book” é a tabela responsável por armazenar as reservas realizadas em locais com proprietário.
- Tabela “chat” responsável por armazenar as mensagens dos utilizadores numa conversa de chat.
- Tabela “userfriendlist” por referenciar os utilizadores que são amigos e os que tem convites pendentes.
- Tabela “notifications” por armazenar as notificações dos utilizadores.

Apêndice C

Páginas da aplicação

Figura C.1 - Página login desktop

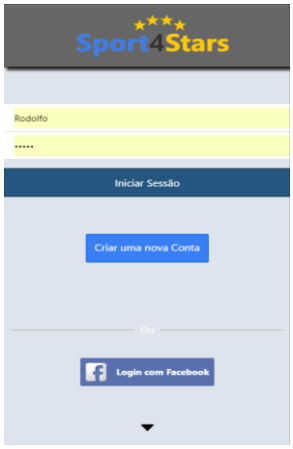

Figura C.2 - Página login mobile

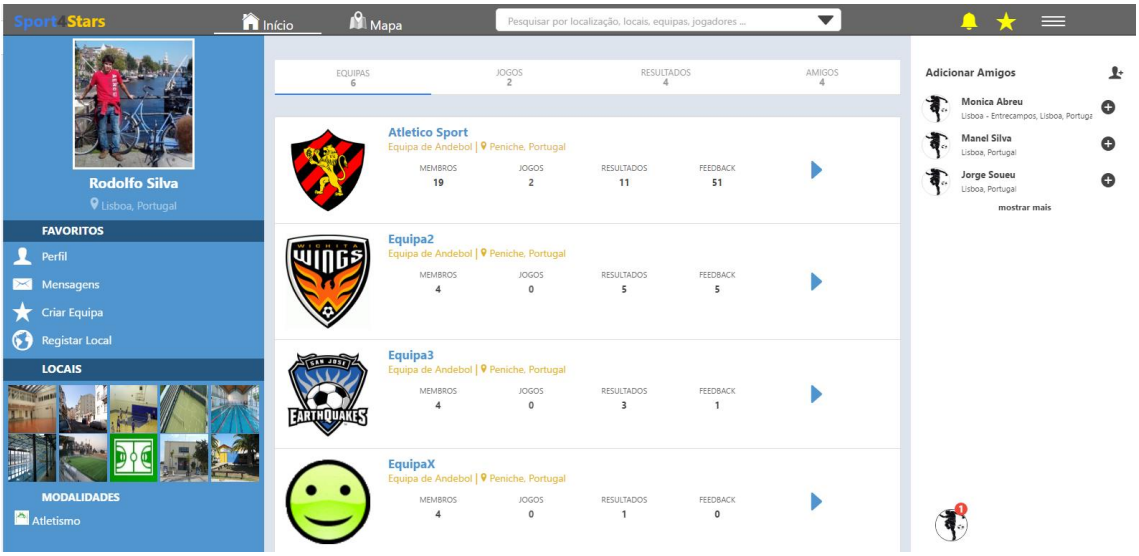

Figura C.3 - Página inicial desktop

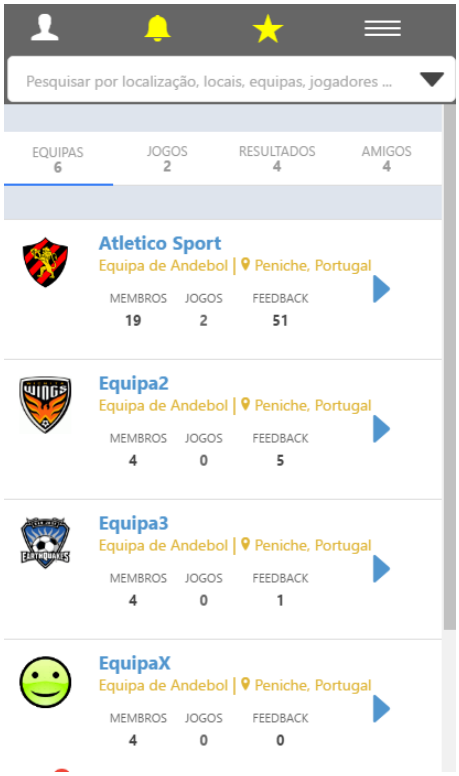

Figura C.4 - Página inicial mobile

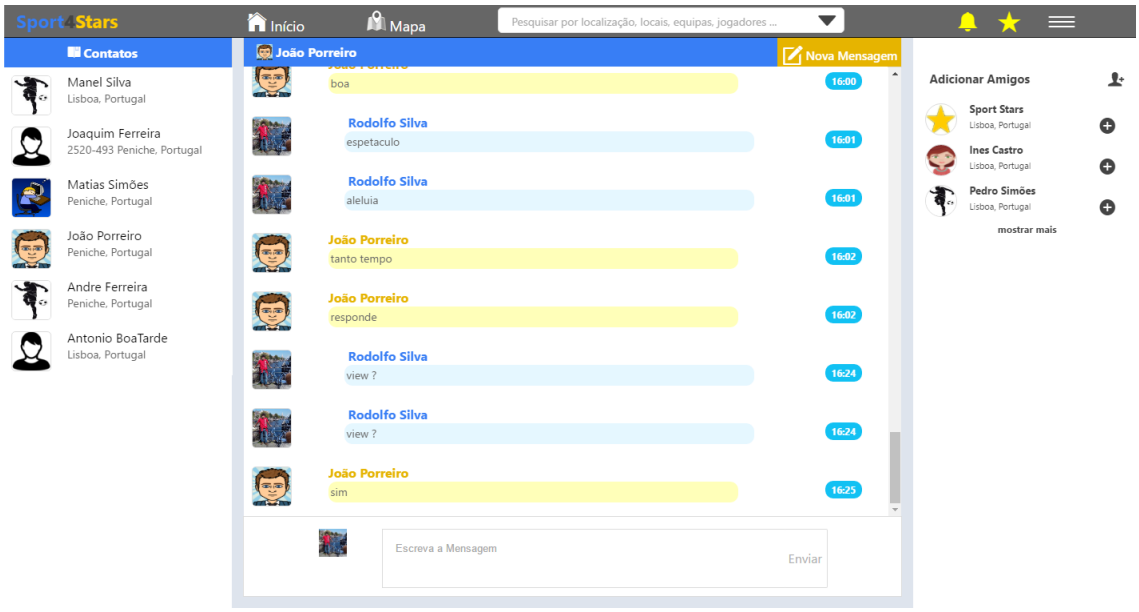

Figura C.5 - Página de mensagens desktop

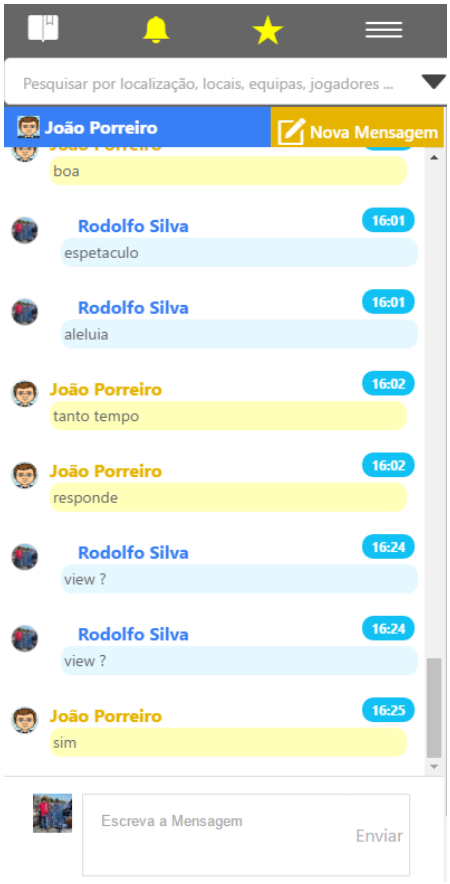

Figura C.6 - Página de mensagens mobile

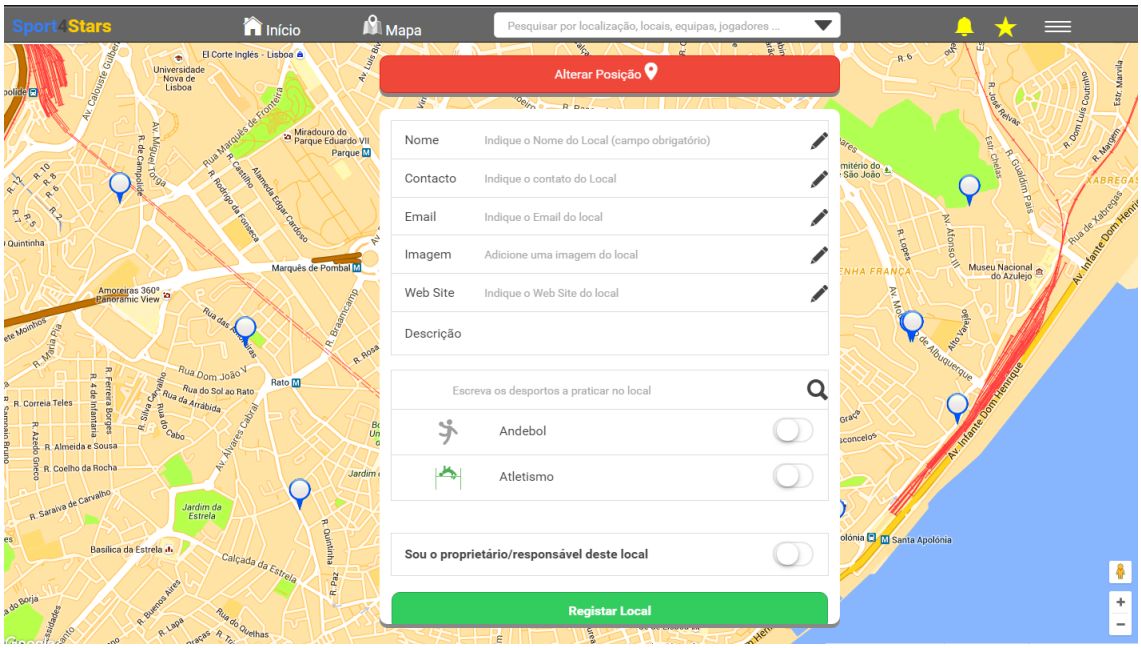

Figura C.7 - Página registar local desktop

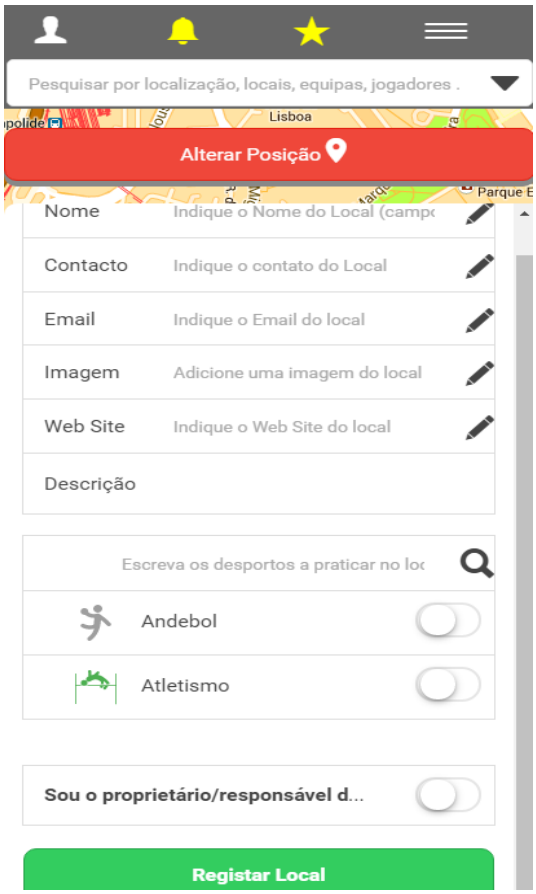

Figura C.8 - Página registar local mobile

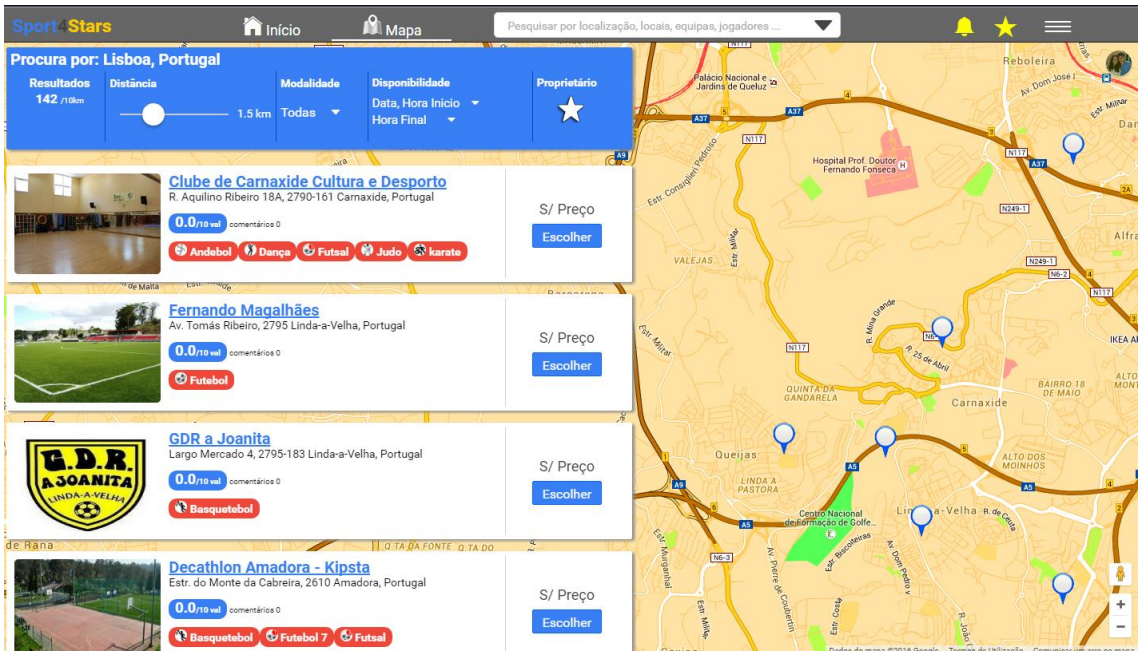

Figura C.9 - Página pesquisar locais desktop

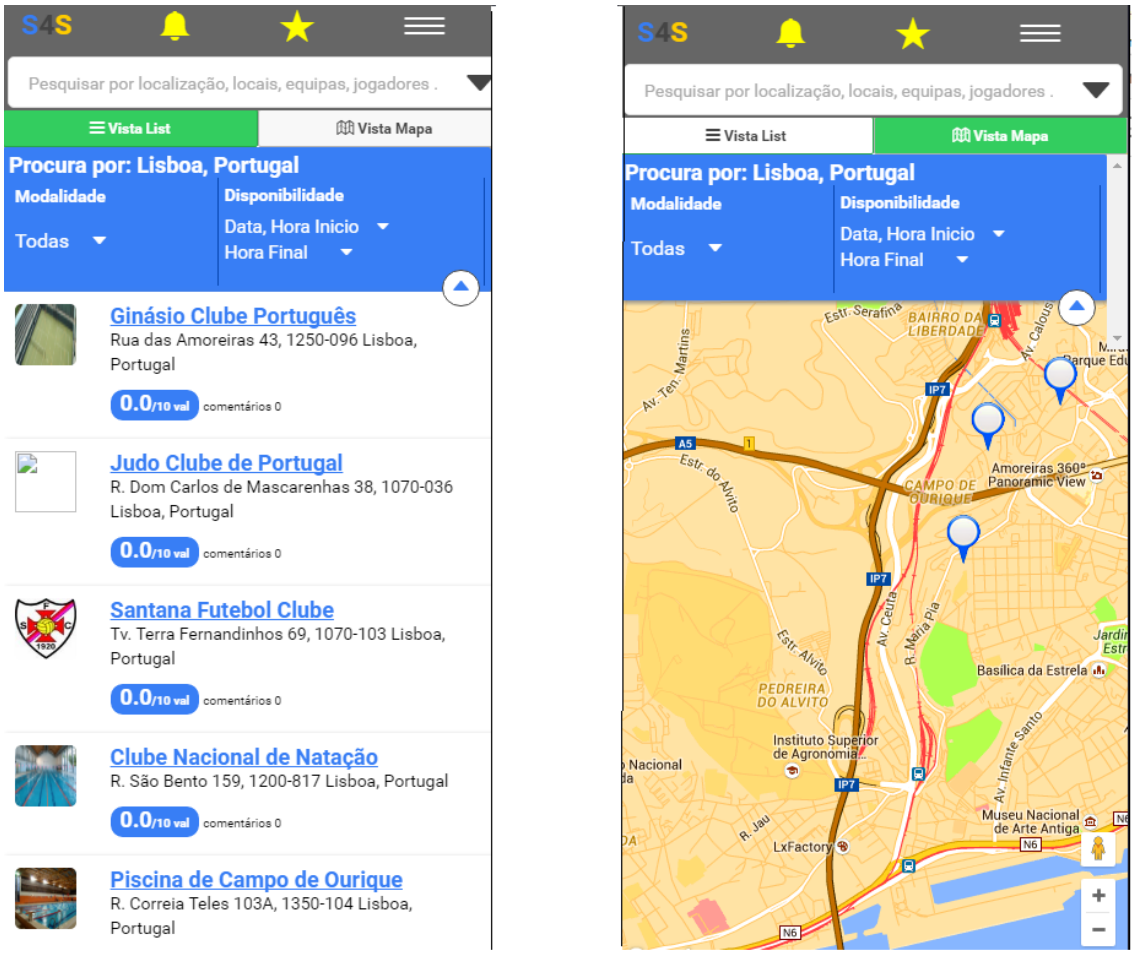

Figura C.10 - Página procura de locais mobile

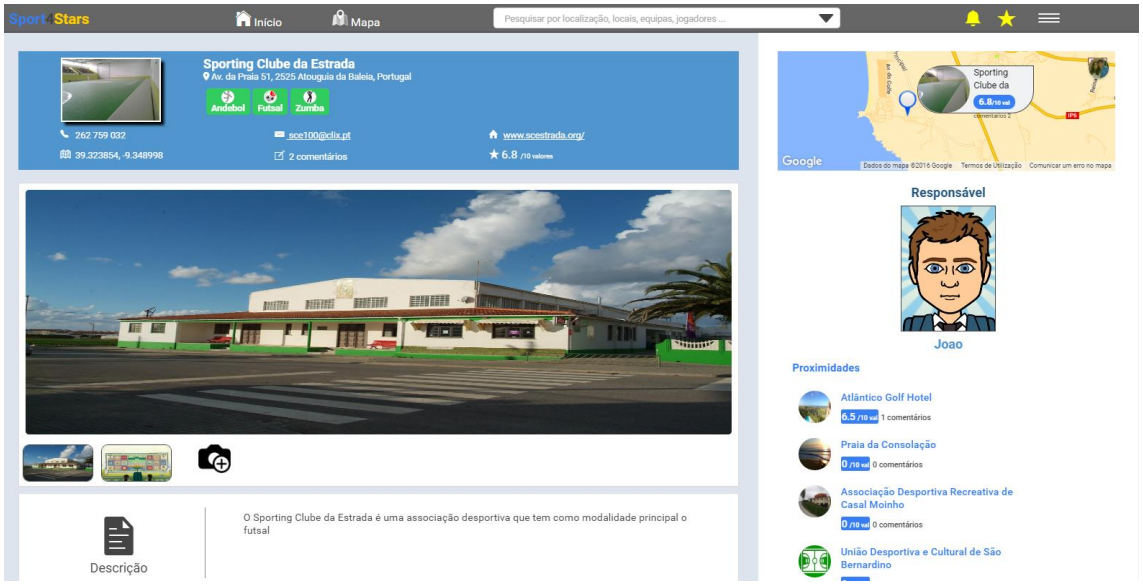

Figura C.11 - Página do local desktop

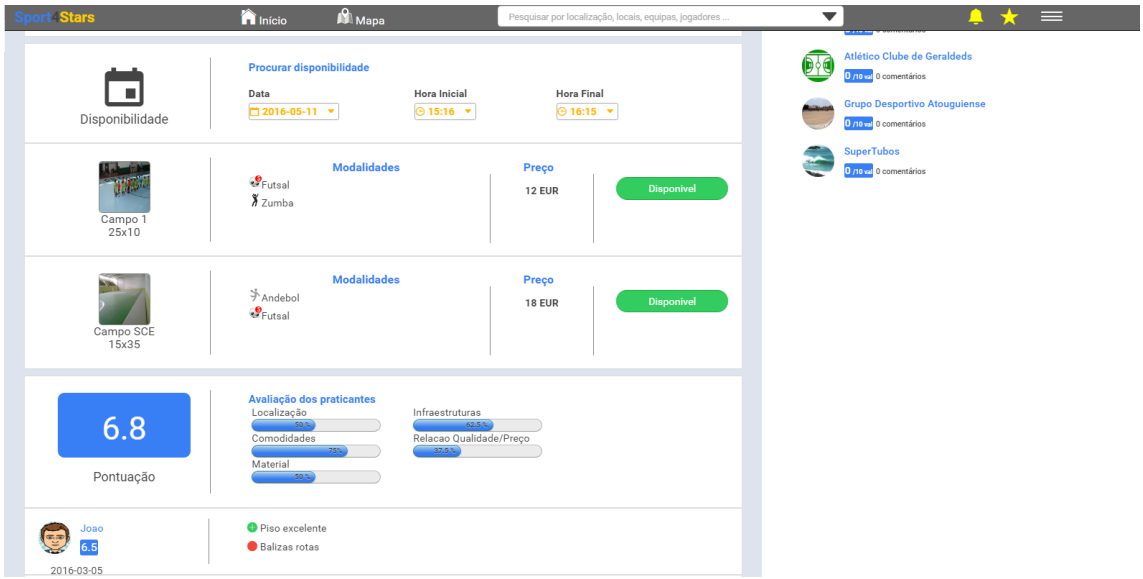

Figura C.12 - Página do local na secção disponibilidade e avaliação desktop

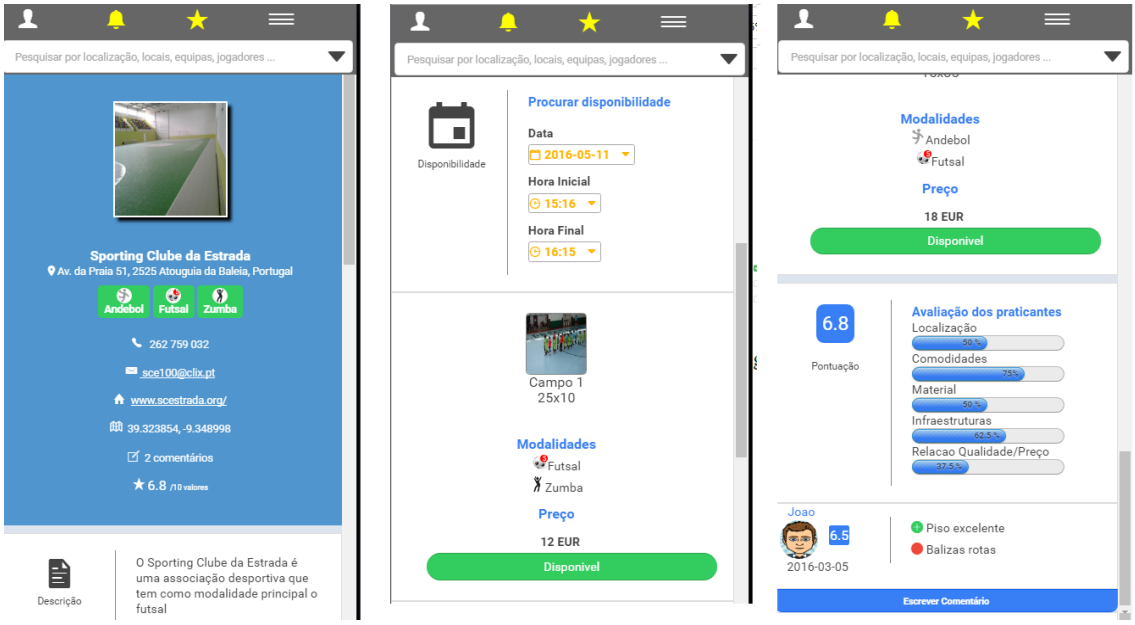

Figura C.13 - Páginas do local em diversas secções mobile

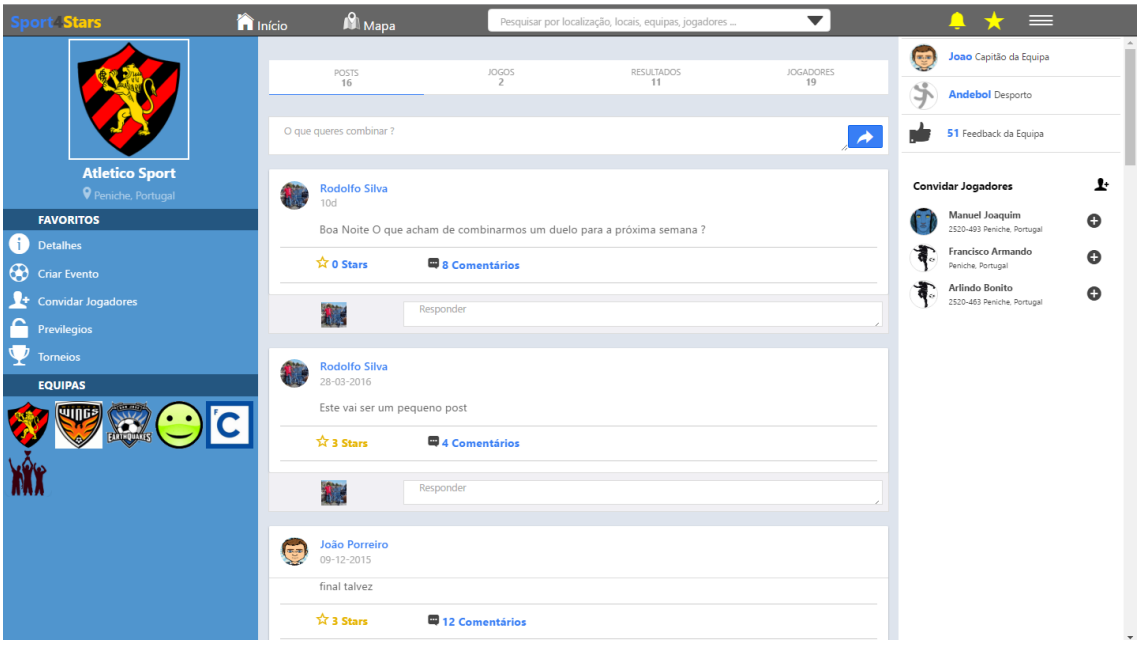

Figura C.14 - Página inicial da equipa desktop

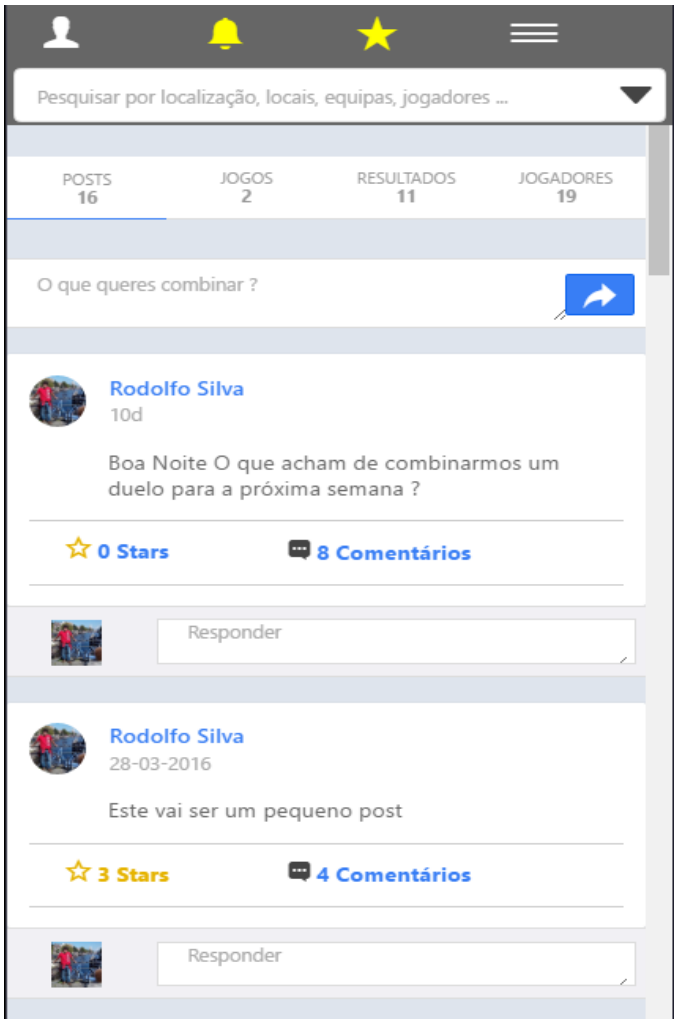

Figura C.15 - Página inicial da equipa mobile

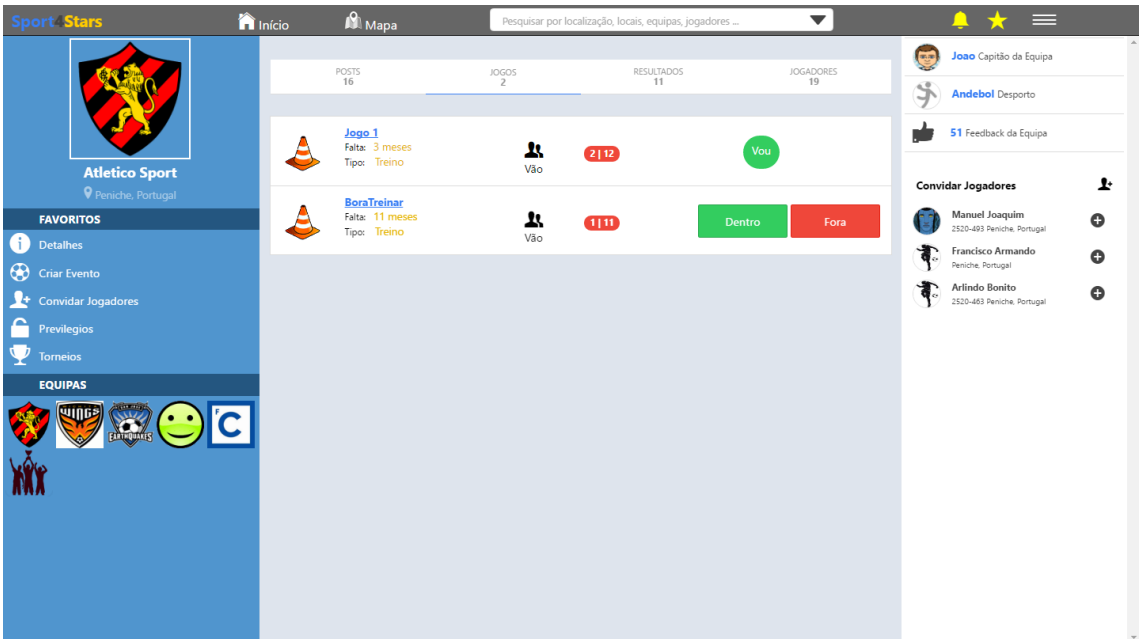

Figura C.16 - Página de jogos da equipa desktop

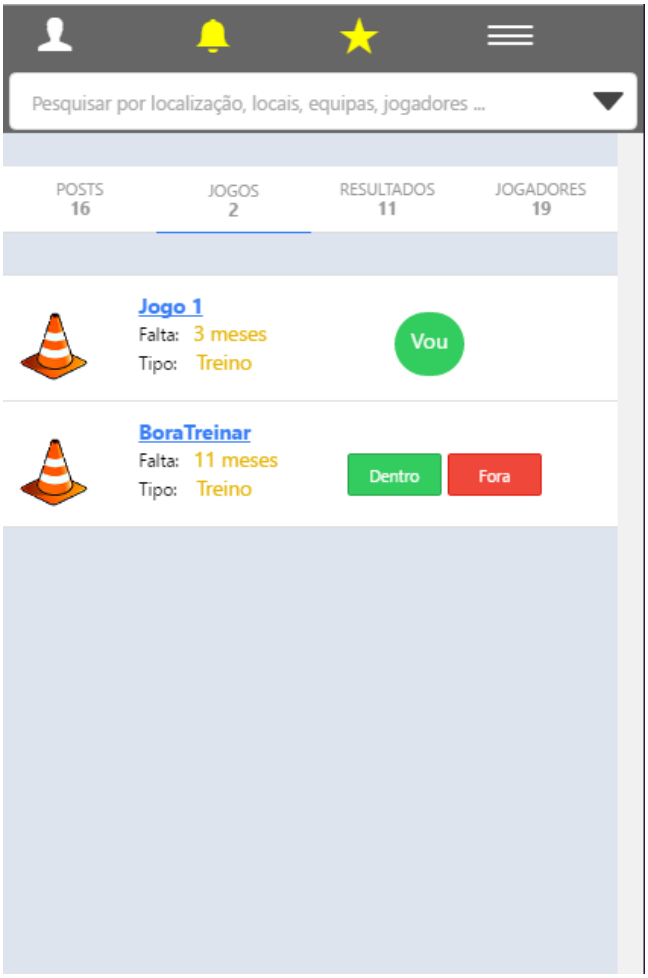

Figura C.17 - Página de jogos da equipa mobile

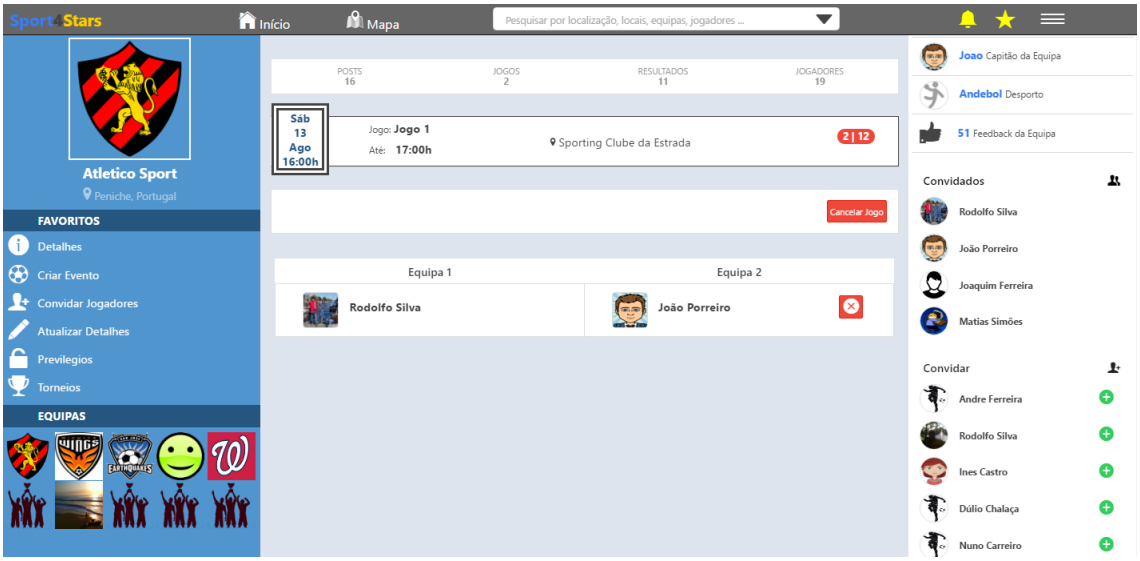

Figura C.18 - Página de treino da equipa desktop

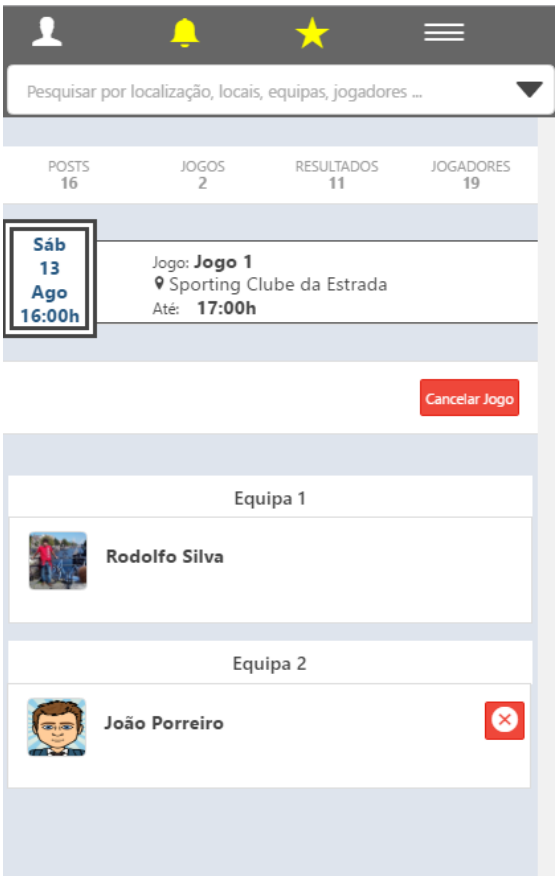

Figura C.19 - Página de treino da equipa mobile

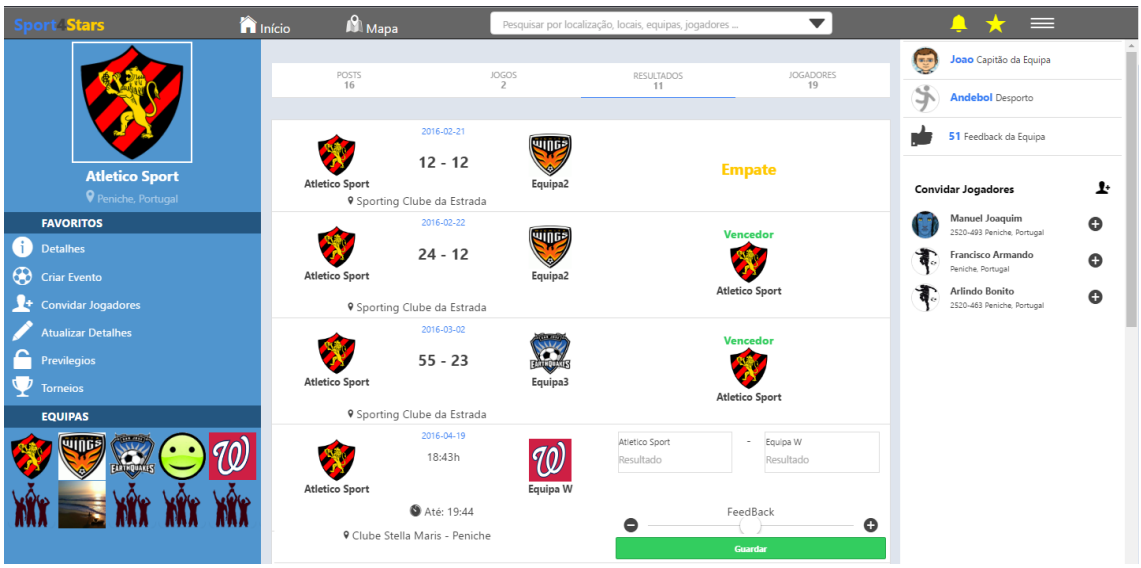

Figura C.20 - Página de resultados da equipa

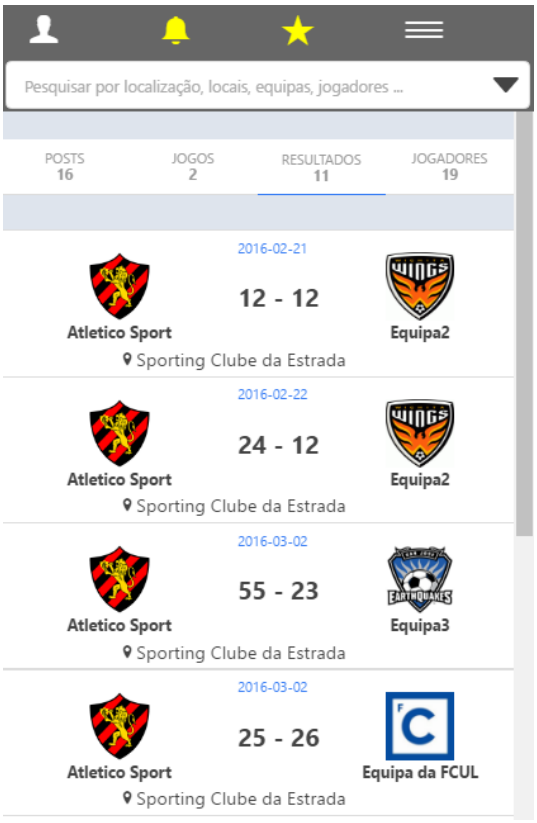

Figura C.21 - Página de resultados da equipa mobile

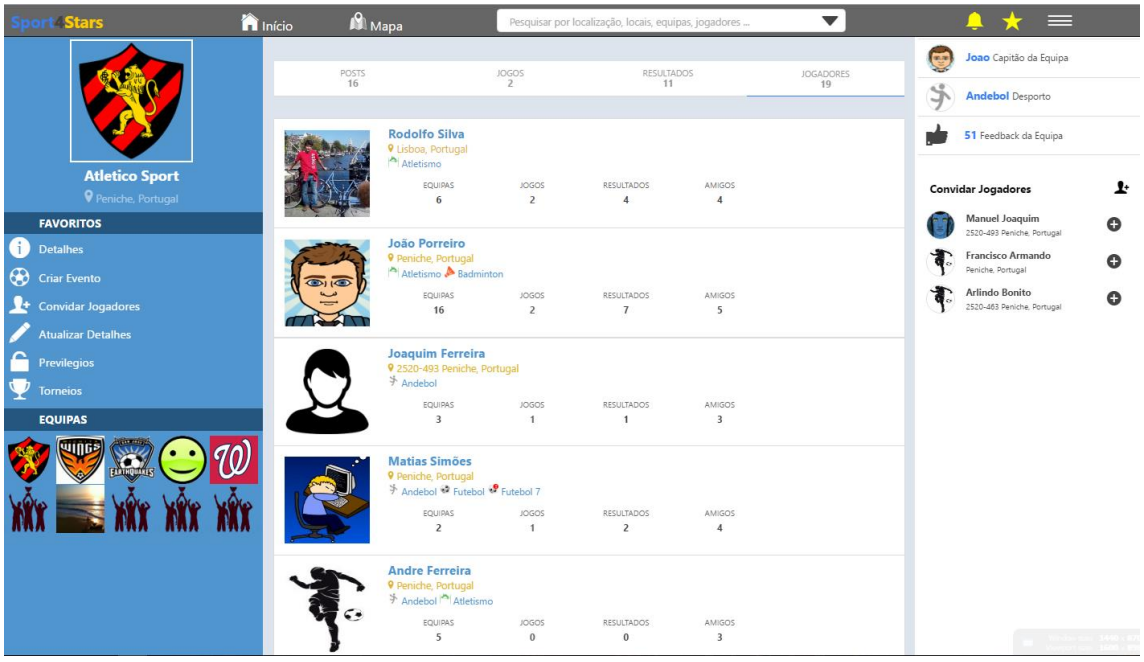

Figura C.22 - Página de jogadores da equipa desktop

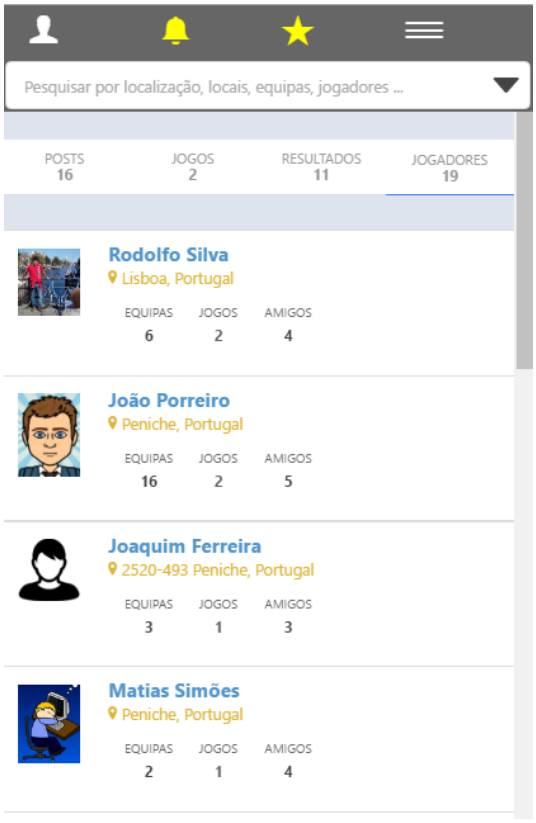

Figura C.23 - Página de jogadores da equipa mobile

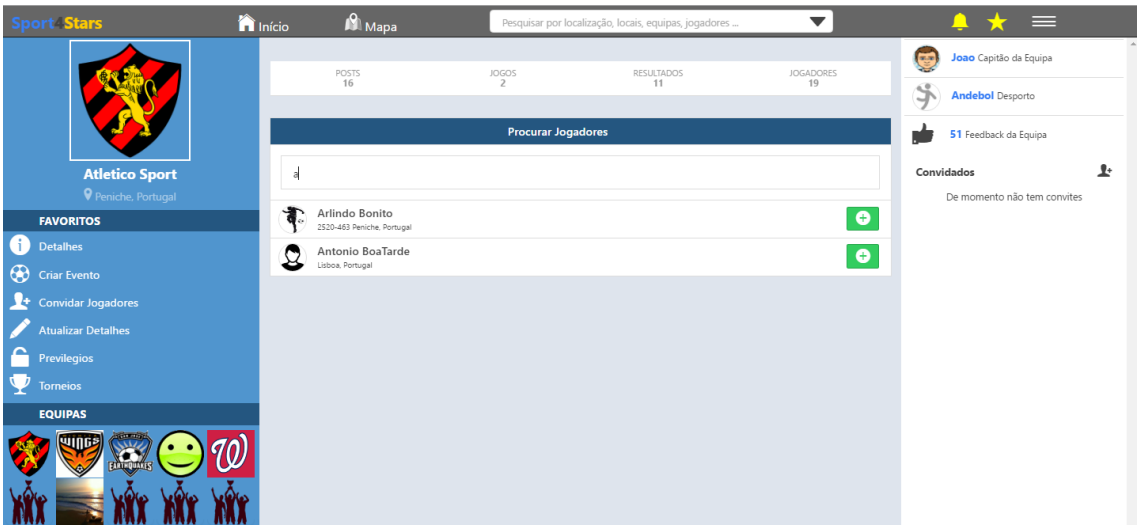

Figura C.24 - Página convite jogadores desktop

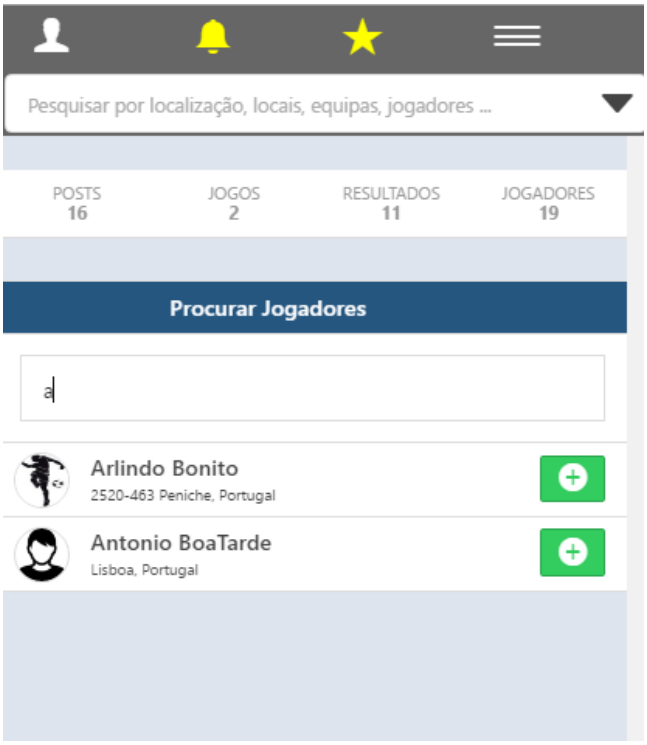

Figura C.25 - Página convite jogadores mobile

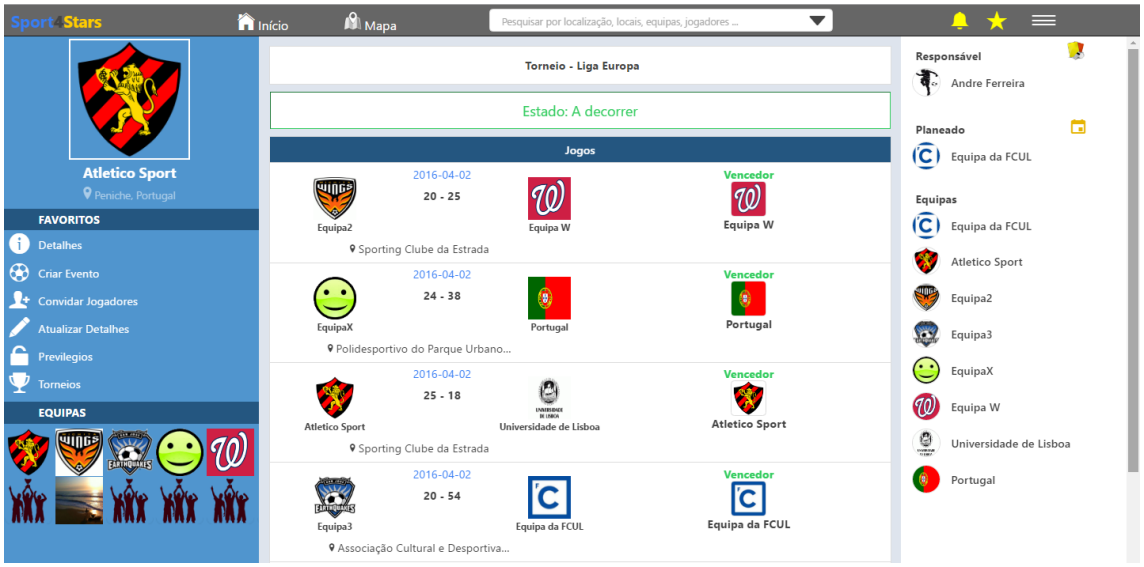

Figura C.26 - Página do torneio desktop

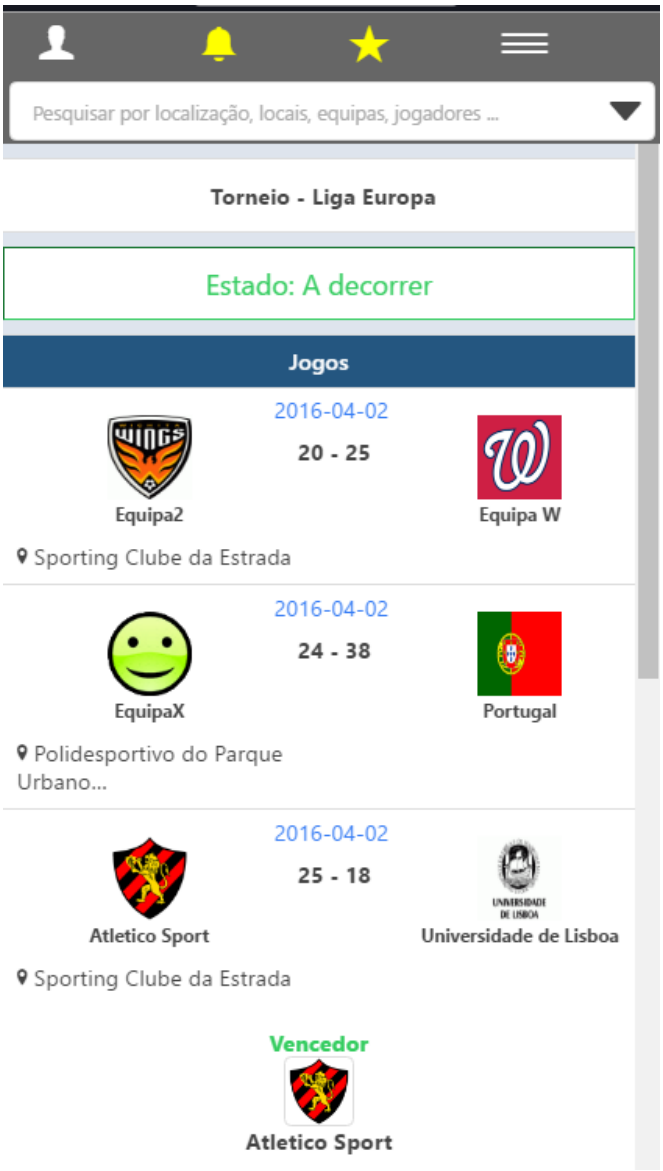

Figura C.27 - Página de torneio mobile

Apêndice D

Tabela com Lista de Funções dos *Models*

Função	**Operação**
insertUser	Responsável por inserir um utilizador na base de dados.
getUserData	Responsável por selecionar os dados de um determinado utilizador pelo seu id.
isFacebookRegisted	Responsável por verificar se o utilizador já está registado como Facebook User.
validate_user	Responsável por verificar os dados de login de um utilizador
getSuggestUser	Responsável por obter sugestões de utilizadores para adicionar como amigo
savePerfil	Responsável por adicionar os dados de um novo utilizador.
updatePerfil	Responsável por gravar os dados atualizados do perfil do utilizador
getAutoTeamComplete	Responsável por obter uma equipa num campo *AutoComplete.*
getResultsGame	Responsável por obter os resultados dos eventos de um utilizador
addFriend	Responsável por enviar um pedido de amizade de um utilizador para outro utilizador

getFriends	Responsável por obter os amigos de um determinado utilizador
isFriend	Responsável por verificar se um determinado utilizador é amigo de outro utilizador.
removeFriend	Responsável por remover a amizade de dois utilizadores.
insertMessage	Responsável por inserir uma mensagem numa conversa de chat.
getMessages	Responsável por selecionar as mensagens de uma conversa de chat.
setMessagesViewed	Responsável por atualizar os estado de visto numa determinada mensagem que já lida no chat.
getSuggestUserPlace	Responsável por obter uma sugestão de locais ao utilizador.
getRefereeTorments	Responsável por obter o utilizador que é responsável por um torneio.
getAutoUserComplete	Responsável por obter uma lista de utilizadores num campo *autocomplete*
getPlaceByID	Responsável por obter os dados de um local por id
insertPlace	Responsável por adicionar um local na base de dados
updatePlace	Responsável por atualizar os dados de um local
updatePerfilPhoto	Responsável por atualizar a fotografia principal do local.
delPhoto	Responsável por apagar uma fotografia da galeria de um local
addPhoto	Responsável por adicionar uma foto a galeria de um local
getPlaceByRadius	Responsável por obter uma lista de locais dado um raio de distância

getAvailabilityRadius	Responsável por obter uma lista de locais disponiveis dado um raio de distância
getNumberOfPlace	Responsável por obter o número de locais dentro de um raio de distância
getPlaceRadiusBySport	Responsável por obter os locais dentro do raio tendo em conta uma modalidade apresentada
getPlacePrice	Responsável por obter o preço mínimo de um espaço num local
daysOff	Responsável por selecionar os dias de folga de um espaço
getAvailableField	Responsável por obter a disponibilidade dos espaços num local
getPlaceFields	Responsável por obter os espaços de um determinado local
getDataField	Responsável por obter a infomação de um determinado espaço.
isDayOff	Responsável por verificar se um determinado dia o espaço está fechado.
getSportsFields	Responsável por obter as modalidades de um determinado espaço.
updateField	Responsável por atualizar os dados de um espaço.
saveDaysOff	Responsável por guardar os dias de folga de um espaço.
saveFieldSports	Responsável por guardar os desportos de um espaço.
updateFieldPhoto	Responsável por guardar as fotografias do espaço.
savePlacesport	Responsável por guardar a informação de um espaço.
bookField	Responsável por reservar um espaço.
updateBookGame	Responsável por atualizar a lista de reservas.
isAvailable	Responsável por verificar se um espaço não foi reservado.
getScheduler	Responsável por selecionar as reservas dos locais.

getBookFields	Responsável por selecionar as reservas de um espaço.
deleteField	Responsável por apagar um espaço.
notReserved	Responsável por retornar se um espaço tem alguma reserva.
updatePhonePlace	Responsável por atualizar o contacto do local.
updateSitePlace	Responsável por atualizar o site do local.
updateEmailPlace	Responsável por atualizar o email do local.
getPlaceAutocomplete	Responsável por obter um conjunto de locais em *autocomeplete*.
getAllSports	Responsável por obter as modalidades da aplicação.
getUserSport	Responsável por obter as modalidade de um utilizador.
updateUserSport	Responsável por atualizar os desportos do utilizador.
saveUserSport	Responsável por guardar os desportos da aplicação.
insertTeam	Responsável por inserir uma equipa na base de dados.
updateTeam	Responsável por atualizar os dados de uma equipa.
addPhoto	Responsável por adicionar uma foto à equipa
getSuggestUserForTeam	Responsável por obter uma sugestão de utilizadores para a equipa
getPrivileges	Responsável por obter os privilégios de um determinado utilizador
getInterestedUsers	Responsável por obter os utilizadores que enviaram um pedido de amizade para a equipa
getInviteUsers	Responsável por obter os utilizadores que foram convidados para equipa
getTeamUsers	Responsável por obter os utilizadores da equipa
deleteTeamInvite	Responsável por rejeitar um convite de um utilizador
confirmTeam	Responsável por confirmar se um membro pertence à equipa
acceptInvite	Responsável por aceitar um pedido de convite

getData	Responsável por obter os dados de uma equipa
insertPost	Responsável por inserir um post na página da equipa
insertStar	Responsável por inserir uma star num post de uma equipa
getStars	Responsável por obter o numero de stars de um post
alreadyStared	Responsável por verificar se um utilizador já inseriu uma star no post
getSinglePost	Responsável por obter apenas um post
insertComment	Responsável por adicionar um comentário a um post
getNumComments	Responsável por contar o numero de comentários de um post
insertNotification	Responsável por adicionar uma notificação
getNotification	Responsável por obter as notificações de um utilizador
isNotificationViewed	Responsável por verificar se uma determinada notificação já foi lida
isViewed	Responsável por obter o número de notificações novas
setViewed	Responsável por atualizar o estado da notificação
savePrivilege	Responsável por salvar o privilegio de um determinado utilizador.
getInvites	Responsável por obter os convites de um utilizador
updateFeedBack	Responsável por adicionar um feedback a uma equipa
getGames	Responsável por obter os jogos de uma equipa
getData	Responsável por obter informação acerca de um jogo
getInvites	Responsável por obter os convidados para o jogo
isBoss	Responsável por obter o jogador criador de um evento
notGo	Responsável por obter os jogadores que não vão a um evento
getNumTeamGames	Responsável por obter o numero de eventos ativos num equipa

getPlaceGame	Responsável por obter o local do evento
getNumInvites	Responsável por obter o numero de convites para o evento
getNumInGame	Responsável por obter o número de jogadores que vão ao evento
getNumInChallenge	Responsável por obter todos os membros que responderam ao pedido de evento
getUsersInTeam1	Responsável por obter os utilizadores na equipa 1
getUsersInTeam2	Responsável por obter os utilizadores na equipa 2
getTeamGamesDone	Responsável por obter os jogos que acabaram
deleteUserGameList	Responsável apagar um utilizador do jogo
getUserJoinedGames	Responsável por obter os utilizadores que estão em algum jogo
insertGame	Responsável por adicionar um treino
isTeamGame	Responsável por verificar se uma equipa está no duelo
createChallenge	Responsável criar um duelo
acceptChallenge	Responsável aceitar um duelo
declineChallenge	Responsável por rejeitar um duelo
getOpponent	Responsável por obter o adversário de uma equipa
insertInvites	Responsável por convidar um conjunto de membros para um evento.
createTorment	Responsável por criar um torneio
insertTormentTeams	Responsável por convidar um conjunto de equipas para um torneio
insertTormentTeam	Responsável por convidar uma equipa para um torneio
updateStatusTorment	Responsável por atualizar o estado do torneio
updateTormentChallenge	Responsável por aceitar um convite para o torneio
deleteInvitedTormentTeams	Responsável por rejeitar um pedido para o torneio
getTormentData	Responsável por obter dados de um torneio

isTormentReferee	Responsável por verificar se o torneio tem um arbitro
getTormentInvitedTeams	Responsável por obter as equipas convidadas para o torneio
getTormentCancelTeams	Responsável por obter as equipas que cancelaram o convite
getTeamTorments	Responsável por obter as equipas que vão jogar no torneio
updateReferee	Responsável por alterar o refere que ainda não aceitou o convite
isTormentFull	Responsável por verificar se o torneio já tem as equipas necessárias
nextTormentPhase	Responsável por obter a nova faze do torneio
getTormentGames	Responsável por obter os jogos dos torneios
getTormentWinner	Responsável por obter o vencedor do torneio
cancelInvite	Responsável por cancelar um convite do jogo
setSuggestTormentGame	Responsável por fazer a sugestão de um jogo no torneio
updataTormentStateGame	Responsável por atualizar o estado do torneio
getDataTormentGame	Responsável por obter os dados dos jogos de um torneio
insertInvite	Responsável por inserir um conjunto de jogadores num jogo
insCapatianAcceptChallenge	Responsável por aceitar um convite para o duelo
getGameChallenge	Responsável por obter os dados de um duelo
isInvited	Responsável por verificar se um jodador está convidado para um evento
getMinElements	Responsável por verificar a equipa que tem menos elementos para um treino
acceptInvite	Responsável por aceitar um pedido para um evento
insertResult	Responsável por adicionar um resultado de um duelo
getDoubleTeam	Responsável por verificar se um utilizador está presente na equipa adversária

updateConfirmGame	Responsável por aceitar o resultado que a equipa adversária deixou
cancelGame	Responsável por cancelar um evento

Tabela D.1 - Funções dos *models*

Apêndice E

Questionário de Testes de Usabilidade

Para realizar os testes de usabilidade foi contruído um formulário online.

Figura E.1 - Descrição da aplicação

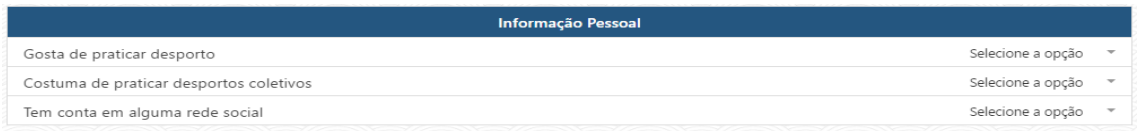

Figura E.2 - Dados pessoais

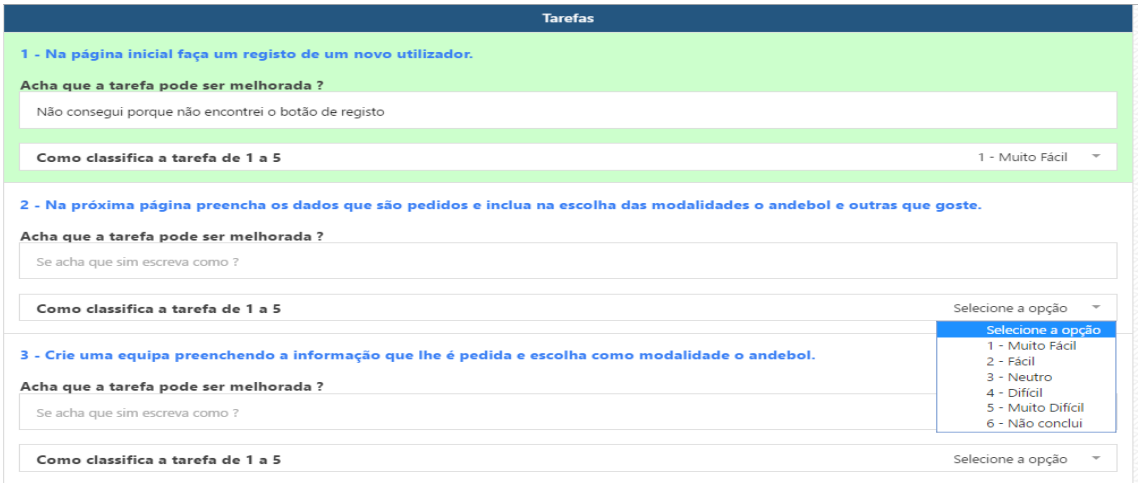

Figura E.3 Tarefa da 1 a 3 com a 1 concluída

4 - Para convidar elementos para sua equipa procure pelo seu amigo "João Porreiro".

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

5 - Faça uma pesquisa na aplicação onde tente encontrar a equipa "Atlético Sport" e entre na página.

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

6 - Envie um convite para a ingressar na equipa.

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

7 - Volte à página de início e entre nas "Mensagens". Envie uma mensagem ao "João Porreiro" a referir que o adicionaste na tua equipa.

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

8 - Aceda ao menu e faça logout na aplicação.

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

9 - Faça um login na conta do João. Em que o utilizador é "joao" e a password é "12345".

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

10 - Visualize as notificações recebidas do João

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

11 - Visualize os convites. Aceite o convite para ingressar na equipa que criou e aceite o convite que enviou para a equipa "Atlético Sport".

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

12 - Responda à mensagem que escreveu anteriormente. "Já aceitei".

Acha que a tarefa pode ser melhorada ?

Se acha que sim escreva como ?

Como classifica a tarefa de 1 a 5 Selecione a opção

Figura E.4 - Tarefa 4 a 12

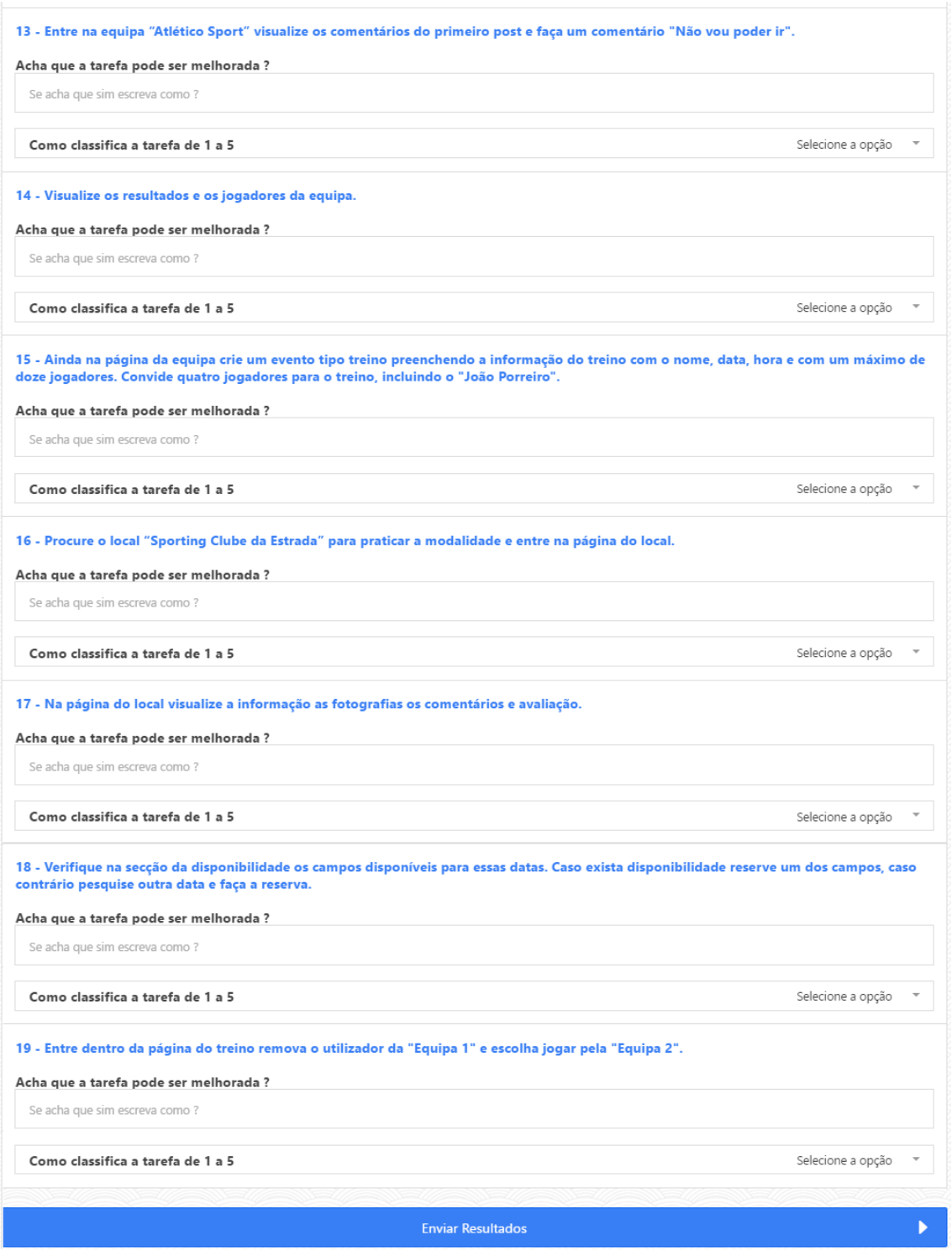

Figura E.5 - Tarefa 13 a 19

Bibliografia

[1] Håkan Nylén. PHP Framework Performance for Web Development Between Codeigniter and CakePHP. (2012)

[2] Albert Kurnia Himawan, Latifah, Hustinawati. Performance Analysis Framework Codeigniter and CakePHP in Website Creation. (2014)

[3] Dr. Lakshmi Prasad Saikia. Comparison of Procedural PHP with Codeigniter Framework. (2016)

[4] Cody Lindley, JQuery CookBook Solutions & Examples for jQuery Developers, pages 21-29, 87-91. (2010)

[5] Louenas Hamdi, Serhan Dagtas. Ajax for Mobility: MobileWeaver Ajax Framework. (2008)

[6] Luís Manuel Rochinha Oliveira. Desenvolvimento de Aplicação Web de Pesquisa, Gestão e Partilha de Eventos. (2014)

[7] Martin Fowler. Patterns of Enterprise Applications Architecture. Addison-

Wesley. pages 21-62 (2003)

[8] Nigel Bevan, ISO and Industy Standards for User Centred Design (2000)

[9] Rolf Molich and Jakob Nielsen. Improving a human-computer dialogue. Communications of the ACM, 33(3):338–348, 1990.